



# WPI

**Interactive Information Console for Atwater Kent  
A Major Qualifying Project  
Submitted to the Faculty of the  
WORCESTER POLYTECHNIC INSTITUTE  
In partial fulfillment of the requirements  
for the Degree of Bachelor of Science**

**Bernardino Garay  
Electrical and Computer Engineering 2017  
Clara Merino  
Electrical and Computer Engineering 2017  
Raphael Swapnil Sarkar  
Electrical and Computer Engineering 2017  
Sonam Sherpa  
Electrical and Computer Engineering 2017**

**Project Advisor:  
Professor Shamsur Mazumder  
Electrical and Computer Engineering Department**

## **Abstract**

There are three major focuses within this Major Qualifying Project (MQP): the upgrade of media displayed on the Atwater Kent Power Panels, the addition of a Smart Mirror, and the relocation of the Independent Grid Solar Charging Station. Changes were made to the features of two past MQP's: Atwater Kent Power Panel (2016) and Grid-Independent Charging Station with Power Flow Display (2012). This will make these projects accessible to everyday users and highlight the awareness of energy usage within Atwater Kent. The Smart Mirror will be able to reduce the amount of power consumed by the Power Panels by not allowing the system to be on for a long period of time. This new combination of systems provides the Atwater Kent community with an interactive method of conveyed energy usage and previous MQP's.

## **Acknowledgements**

This project was made possible by the New England Center for Analog and Mixed Signal Integrated Circuit Design (NECAMSID) and by the Electrical and Computer Engineering (ECE) Department at WPI. The team would like to thank Professor Shamsur Mazumder for their guidance throughout the course of this project. The team also would like to thank Professor O'Rourke for advice on how to approach the project's proposals. The team would also like to thank Bill Appleyard and Leah Morales for their assistance in ordering parts and assembly. The team would also like to thank Professor Orr for inviting the team to present at the Energy Sustainability Forum. The team extends their gratitude to the Head of the ECE Department, Professor Massoud for providing assistance to implement new features to the existing system featured in the front lounge of Atwater Kent.

## Table of Contents

<b>Abstract</b> .....	2
<b>Acknowledgements</b> .....	3
<b>1. Introduction</b> .....	6
<b>2. Preliminary Project Decisions</b> .....	6
<b>2.1. Related Previous Projects</b> .....	7
<b>2.1.1. Atwater Kent Power Panel</b> .....	7
<b>2.1.2. Renewable Energy Applications</b> .....	8
<b>2.1.3. Grid Independent Solar Charging Display</b> .....	9
<b>2.2. Atwater Kent Power Panel Design</b> .....	11
<b>2.2.1. LED Panel Matrix</b> .....	11
<b>2.2.2. LED Wall Controller</b> .....	12
<b>2.2.3. Capacitive Touch Sensor</b> .....	13
<b>2.2.4. ODROID XU4</b> .....	15
<b>2.3. Project Surveying for Deciding on Enhancement</b> .....	16
<b>2.3.1. Previous MQP Group Responses</b> .....	16
<b>2.3.2. Students</b> .....	16
<b>2.3.3. Parents &amp; Prospective Students</b> .....	16
<b>2.3.4. Professor Responses</b> .....	17
<b>2.4. Final Project Decisions</b> .....	18
<b>3. System Upgrades and Additions</b> .....	20
<b>3.1. Challenges of Operating the Previous System</b> .....	20
<b>3.1.1. Electrical Component Enclosure</b> .....	23
<b>3.1.2. Capacitive Touch Sensors</b> .....	26
<b>3.2. Power Panels Media Upgrade</b> .....	29
<b>3.2.1. Children Panels</b> .....	29
<b>3.2.2. Main Display Selection Menu</b> .....	31
<b>3.2.3. Challenges</b> .....	33
<b>3.3. Solar Grid Independent Charging Station</b> .....	39
<b>3.3.1. Current System and Goal</b> .....	39
<b>3.3.2. Procedure</b> .....	41
<b>3.3.3. Finalization of System</b> .....	45
<b>3.4 MQP addition Criteria</b> .....	47

<b>3.5</b>	<b>Smart Mirror</b> .....	48
3.5.1.	Block Diagram and Component Analysis .....	48
3.5.2.	Installation Process .....	52
3.5.3.	Smart Mirror Features .....	57
<b>4.</b>	<b>Future Recommendations</b> .....	58
4.1.	LED Panels .....	58
4.1.1.	Electrical Component Enclosure .....	58
4.1.2.	Games.....	59
4.1.3.	Solar Panels Installation .....	59
4.2	Smart Mirror.....	60
4.2.1.	Remote Desktop.....	60
4.2.2.	User Control .....	60
4.2.3.	Lab Occupancy.....	61
<b>5.</b>	<b>Conclusion</b> .....	62
<b>6.</b>	<b>Appendix</b> .....	63
6.1	Power Panels User Manual .....	63
6.2	Making System autonomous .....	66
6.3	ODROID and Software Tips.....	69
6.4	Accessing Advanced User settings .....	71
6.5	LED Studio Software Manual.....	72
6.6	System Power Issues .....	76
6.7	Smart Mirror User Manual .....	78
6.8	Code from Smart Mirror .....	80
6.9	Code from LED Panels .....	128
6.9.1	Power Panel Section.....	128
6.9.2	Children Panels Section.....	140
<b>7.</b>	<b>References</b> .....	147

## **1. Introduction**

There are three major focuses within this Major Qualifying Project (MQP): the upgrade of media displayed on the Atwater Kent Power Panels, the addition of a Smart Mirror, and the relocation of the Independent Grid Solar Charging Station. Before the team could tackle these three focuses, the primary goals were to develop an understanding of how the previous MQP's functioned and their intended goals, to improve upon the documentation from in the MQP report provided by the Power Panel MQP made in 2016, and to develop a list of media that the Atwater Kent community would enjoy interacting with. The group identified that there was a need to improve upon documentation because there was a lack of data needed to understand the overall system. This was highly needed before being able to move ahead and add more features. The report provided by the Power Panels team contained information that was unclear about system installation and functionality. However, the Power Panel MQP group did install and set up the entire Power Panel system which provided a method of displaying information relevant to students and visitors onto an aesthetically appealing LED display matrix. Since then, the Electrical and Computer Engineering (ECE) department has provided the 2016-2017 team with ideas, resources, and funding that allows and encourages the team to build upon the existing system with new features and identify occurring flaws. This report will touch upon what how previous MQP's are enhanced and tied together to make an energy usage awareness system for the Atwater Kent community.

## **2. Preliminary Project Decisions**

One of the goals of this project is to provide a platform to display energy usage related statistics. The Power Panels serve as a tool to add character to the lounge as well as a way to display information gathered from a variety of previous MQP projects. It was necessary to research previous projects completed by WPI students related to energy to see how they contributed to the Atwater Kent Power Panels. Each of these systems is explored further through this section.

## 2.1. Related Previous Projects

### 2.1.1. Atwater Kent Power Panel

This is an analysis to understand what the 2015-2016 Power Panel Group accomplished. For the remainder of this report, the team will be referring to this MQP as the previous MQP team. The goal of the previous MQP team was to design and assemble a Power Panel system that displays relevant information as well as highlights energy consumption by Atwater Kent to students and visitors of the Atwater Kent Pumpkin Lounge. Their panel displayed a variety of information to students which included campus events, a twitter feed, campus map, pizza countdown, solar data measurements, and the standard weather and time. A creative element was explored through analyzing and comparing various LED pixel layouts to make the system more aesthetically pleasing. In addition, capacitive touch sensors installed on the wall, in front of the panel displays, allow for users to interact with the system's menu selections.

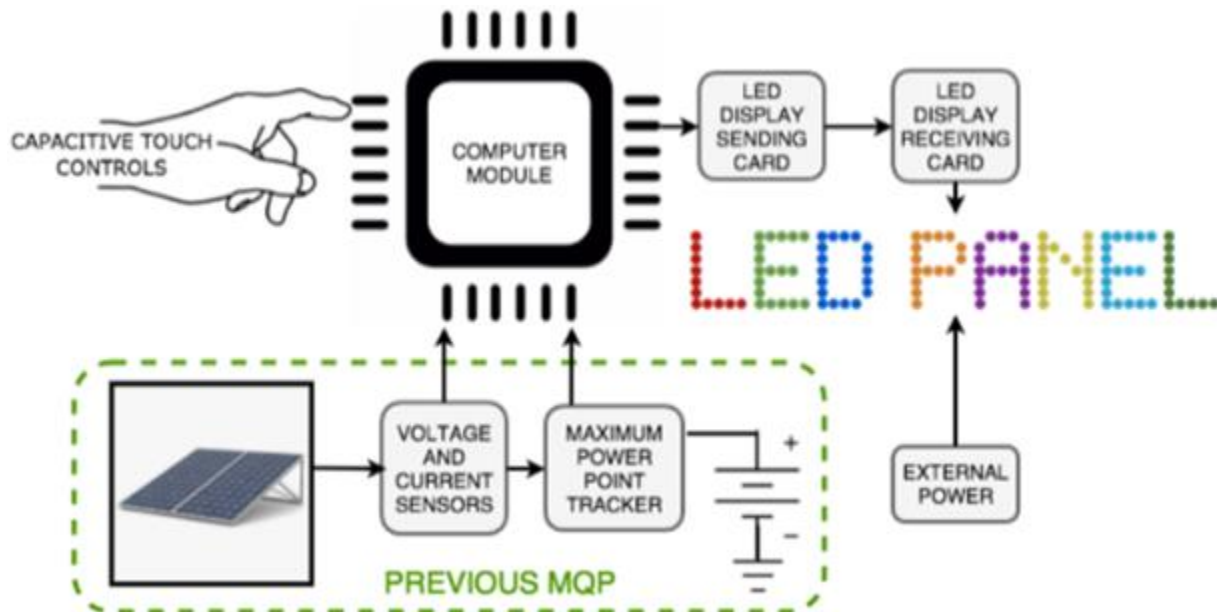


Figure 1. Previous MQP Project: Atwater Kent Power Panel System Block Diagram

The block diagram above presents the overall Power Panel system design. The system utilizes another MQP (Grid Independent Charging Display) in order to receive power generated from the solar panels. The voltage and current sensors from the

Charging Display serve as input to the computer module of the system to display solar panel statistics and data. The user is able to navigate the system through the capacitive touch sensor that directly inputs into the computer module. The computer module outputs to an LED display sending card which communicates with the LED display receiving card. The receiving cards connect to the LED matrix panels in order to mirror the screens displayed on the computer module. The computer module is powered through an external power supply.

### **2.1.2. Renewable Energy Applications**

In 2012, this team evaluated the extent to which using power obtained from the wind turbine was feasible. After determining that the wind turbine did not create enough power to use for their Power Panel, they created a “solar energy harvesting board” (3). This board served to store energy in batteries for use within the ECE department.

Initially, this team had three sections of renewable energy applications which they focused on. With interest in developing a solar panel system, wind system, and site monitoring system, it was determined that in order to stay on schedule, the solar panel system was explored. The roof of AK was surveyed and the most optimal location for the solar panels was determined. The team installed three out of the six panels that were donated to the NECAMSID Lab. The remaining four have not yet been installed and are still located in the Lab. The solar panels located on the roof are still functional and will be utilized by this Power Panel project for the Atwater Pumpkin Lounge. An image of the set-up on the roof of AK can be seen in Figure 2 with the solar panel installation circled in red.



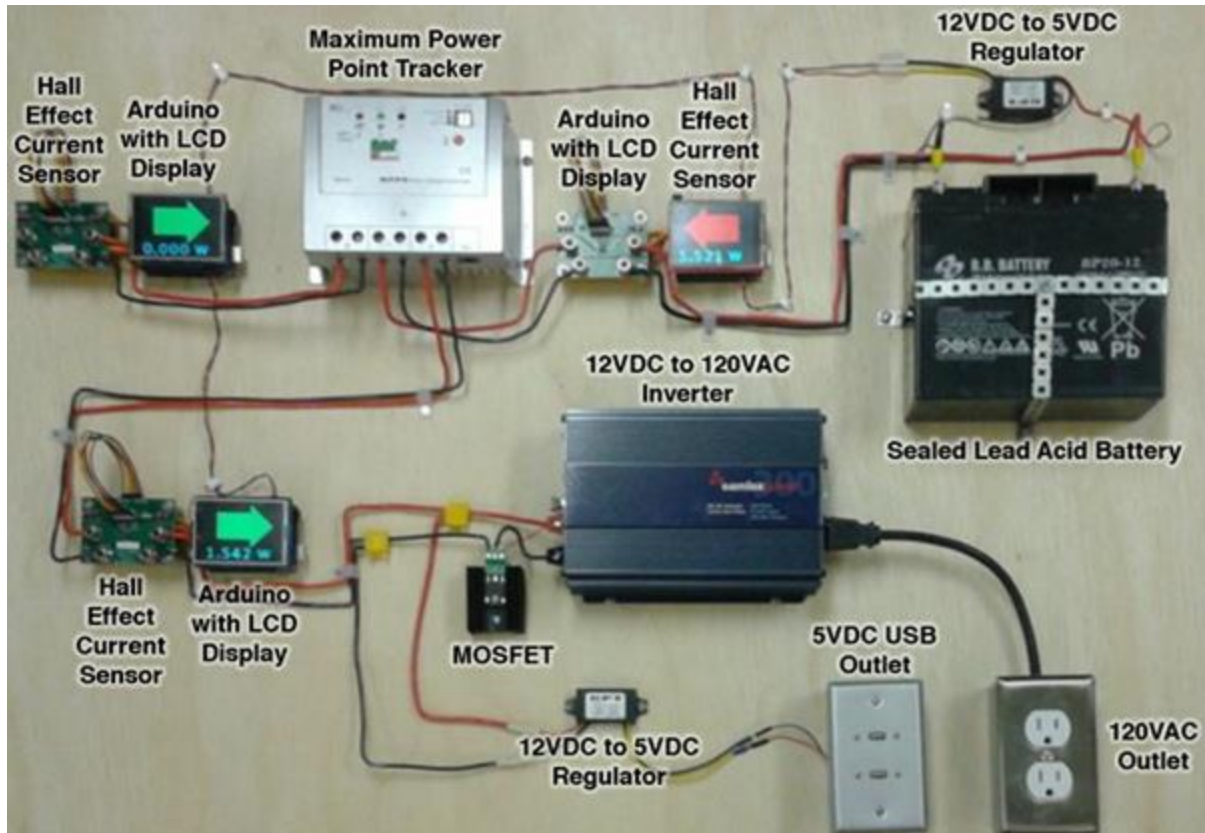


*Figure 2. Location of Solar Panels on roof of Atwater Kent*

The 2012 team proposed to install a panel display similar to the intentions of this Power Panel. The past team focused the majority of available time on establishing an effective method to collect energy from the panels and therefore was not able to develop a means for display. The project concentrated on developing a system to measure the output voltage and current accurately. This has been used to collect and make assessments of the output energy from the solar panel array.

### **2.1.3. Grid Independent Solar Charging Display**

In 2012, the MQP, “Grid-Independent Charging Display,” was developed to display the power flow of the AK solar panels into a wall mounted charging station (5). The primary goal of this project was to promote the use of green energy while also providing a useful device to the visitors of Atwater Kent. Figure 3 shows an image of their final design.



*Figure 3. Grid Independent Solar Charging Display System Diagram*

The project displays the power flow at three different points: the solar panel output, battery charging circuit input, and the load output. This was meant to give the user a logical understanding of how power flows from generation to consumption. By continuously displaying data through LCD screen modules, the team expected users to become more conscious about their energy usage and needs.

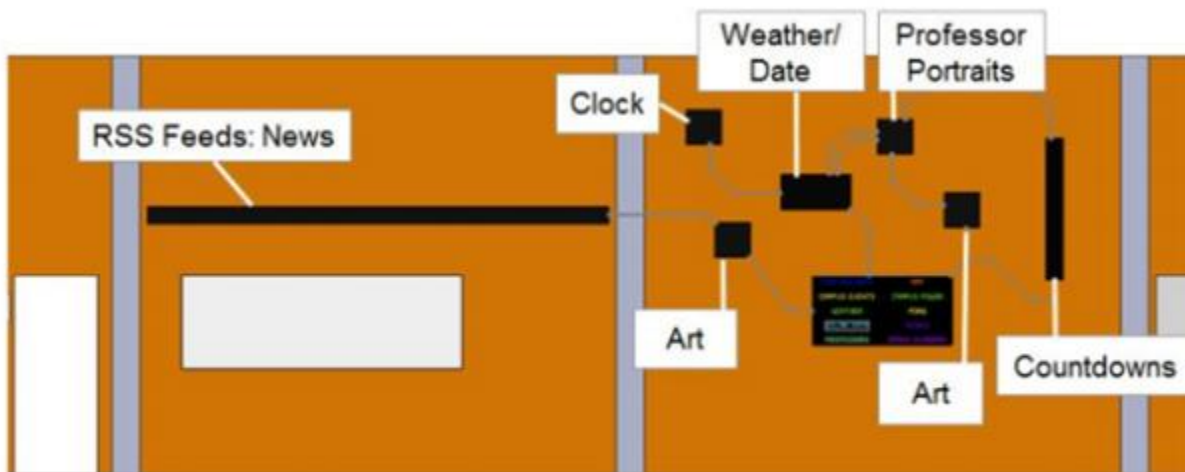
Instead of being installed in the Pumpkin Lounge, the project remained in the NECAMSID lab where it was occasionally used. It remained there for a number of years until an issue with one of its LCD displays arose and could not be easily fixed. Additionally, the previous MQP team came up with some improvements that they felt could be easily implemented into a future redesign. The team felt that one of the major concerns with the design was the limited amount of space for displaying text and images on the 2.8-inch LCD touch screens (5). Initially, they wanted to display the energy saved

in terms of equivalencies such as kilowatt-hours saved, joules of energy from the sun, or amount of money saved. They recommended incorporating larger display screens at each point or relaying the information from the Arduinos via Ethernet to a TV panel in order to display more visually appealing images and text.

## 2.2. Atwater Kent Power Panel Design

### 2.2.1. LED Panel Matrix

The previous MQP team opted to work with LED panels rather than a LCD screen because of the flexible design and to showcase electrical ingenuity. The LED panels can be joined into a matrix and made into any size that they want while an LCD screen usually has a standard size. This decision provided more freedom to the Panel design which the previous team has made in varying sizes and shapes to add to the artistic value. As the goal of the project was also to show the creative ingenuity of WPI students, the LED panel matrix that they designed, built and installed in the building is a lot more impressive than using LCD screens and just using it as a monitor.

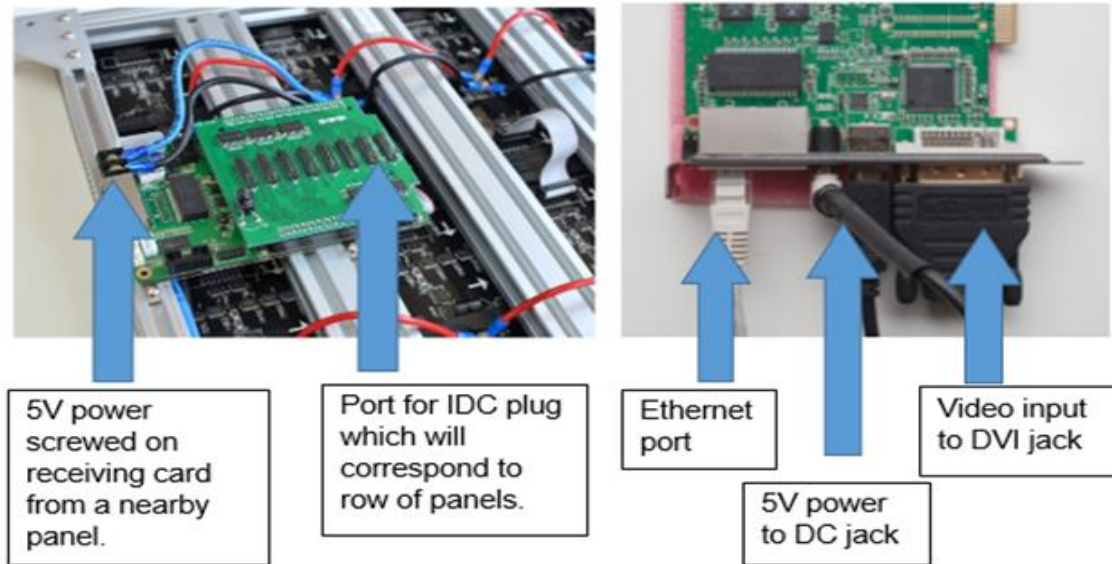


*Figure 4. Previous System Display Decisions*

### 2.2.2. LED Wall Controller

In the final LED panel design, the 2016 team used LED Video Wall controller from Adafruit to display the video on the panels. The system involved video decoder boards with a sender card connected to the video source and receiver card connected to the LED wall panels. Sending card is located inside the power box while the receiver card is located behind the main display panel in the pumpkin lounge. The two cards communicate over Ethernet. For compatibility with the video decoder boards, 16X32 LED panels used were also from Adafruit as the software configuration and the rest of the tutorial were modeled with those panels. Adafruit has a tutorial on setting up LED Video Wall on their website which walks through the various steps in building the frame, wiring/mounting the system and LED configuration software. The previous team were working on the setup of the system until C term which gave them little time to document a lot of their progress and design motives. Additionally, while the LED Video Wall Controller is key to the foundation of this project, Adafruit explicitly states that they don't have any additional documentation other than the tutorial page and that they don't provide any consultation or support for the product.

The figure below shows the different connections of the video decoder boards for them to work properly. The images were pulled from the tutorial page to give an idea of how the sending/receiving cards look and their connections.



*Figure 5. Wiring of Receiver Card in the LED Panel (Left) Sender Card Connections (Right)*

### 2.2.3. Capacitive Touch Sensor

To navigate the system, they built and installed a capacitive touch sensor. A capacitive touch sensor uses a conductive material that changes capacitance when touched by a grounded object such as a hand. Capacitive buttons can be labeled and laid out in intuitive manner. Some capacitive touch specific solutions exist on the market, but nearly any microcontroller with analog input pins can be used to detect capacitive touch. The Arduino microcontroller has libraries that support capacitive touch recognition with the benefits of low cost and ease of programming. The touch pads themselves can be manufactured out of any conductive material that has excellent hardness and strength properties compared to the human hand. Additionally, the pads can be designed in a custom layout that fits the Power Panel's user interface and can be easily interfaced through an Arduino to the central processor. Because of these positive aspects, a capacitive touch interface was utilized for the user interaction portion of the Power Panel.

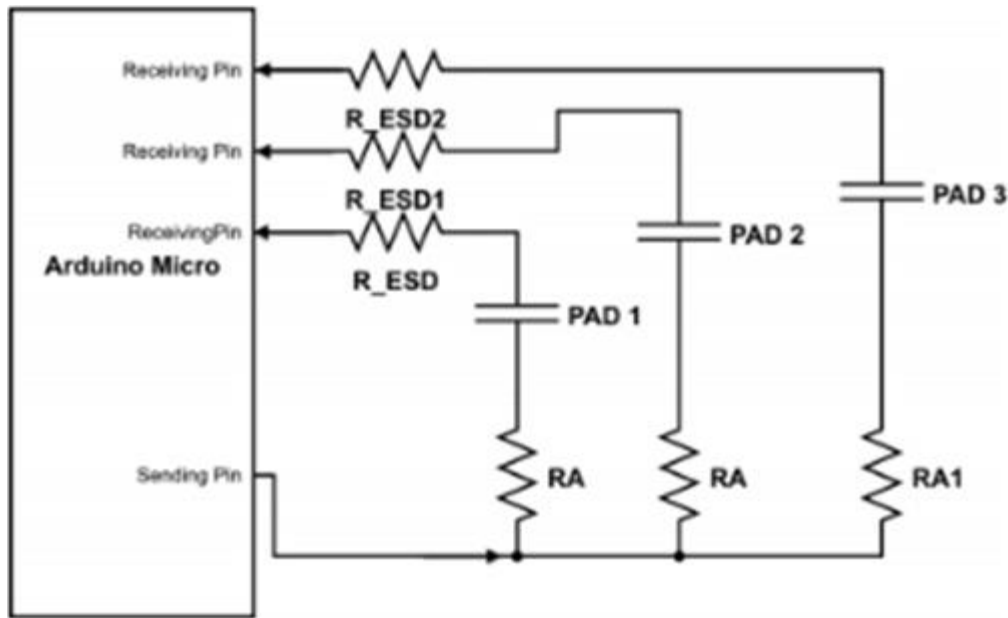


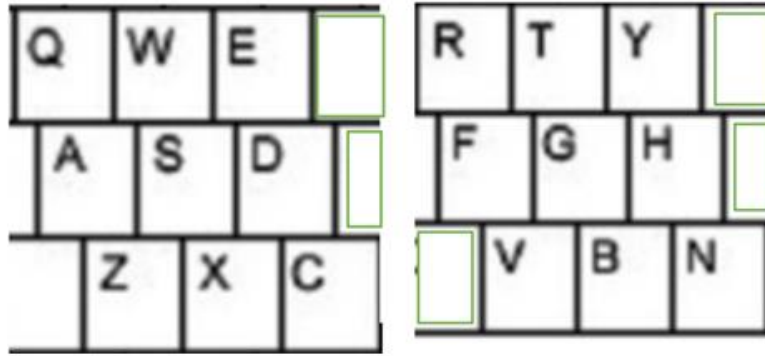
Figure 6. Capacitive Touch Sensor connected with Arduino Micro

They designed it to be intuitive and simple to use with the main display. It was the only user input to the Power Panel so it was responsible for controlling the menu options and programs within. One of the better display of the capacitive sensors was its use to simulate a game of pong displayed on the screens. Although relatively simple, it allows for 2 players to use the sensors to play the game back and forth, demonstrating the interactive capability of the system. Among the 9 pads, the triangle and square pads at the bottom were left without any assigned functions for increased flexibility in future works with the capacitive touch sensor.



Figure 7. Capacitive Touch Sensor mounted in doorway





*Figure 8. Capacitive Sensors output to ODROID from left and right panel*

The pictures above (Figure 7 and Figure 8) show the capacitance touch sensors mounted on the doorway and their respective output that gets read by the ODROID. During the testing phase, you can use the keyboard connected to the ODROID instead of the capacitance sensors for debugging.

#### **2.2.4. ODROID XU4**

The previous team needed a computer module capable of integrating the LED Matrix panels, LED Wall Controller and capacitive touch sensors into one complete functional system. Based on their criteria, the microcontroller chosen was the ODROID XU4, which runs a Cortex A-15 Processor with a clock speed of 2GHz. The ODROID XU4 had many of the desired features such as decent processing power, digital I/O pins, Analog to digital converters, DVI video output, memory and operating system capable of graphic signal output. It also ran a more developed version of the Ubuntu OS allowing for simple installation of programs such as Processing which is the main program used to control the display on the LED panels. It also minimizes the amount of time spent on editing the operating system's settings to allow for dependable installation. From all the considered options, the ODROID XU4 had the highest clock speed enabling it to run Processing very smoothly compared to some of the other options such as raspberry pi and Arduino.

## **2.3. Project Surveying for Deciding on Enhancement**

### **2.3.1. Previous MQP Group Responses**

Throughout the summer as well as during A-term, the team kept in close contact with the previous MQP group. This was to learn about the system but it also helped with learning what they saw as possible system improvements. During a meeting with Alejandro Miranda, the team received feedback on how to improve the system through areas of troubleshooting.

### **2.3.2. Students**

To approach students, the team turned on the system and provided a google survey for passersby to fill out. The system itself attracts many passing by and so collecting a wide range of responses became feasible to accomplish. The team conducted surveys for approximately 15 hours total. To reach out to more students that were not available during the demo, the team also sent out an email to the undergraduate population. This email had a link to the google survey form. The google survey was short and provided images of what the current system looked like. It asked students to provide feedback on what is currently being displayed and provide interest rankings of suggested system added features. It also had a textbox for students to add any suggestions or comments as well as indicating interest in providing assistance with the panels themselves. This was done to provide the possibility of having a group of students who can be system operation managers after the completion of the MQP.

### **2.3.3. Parents & Prospective Students**

During Parent's Weekend, there was an opportunity to collect survey results from prospective students as well as parents. By interviewing parents, the team received feedback from an audience that was not technically inclined. This provided unique suggestions on how to attract a crowd outside of the Atwater Kent community audience. Interviewing prospective students help the team add onto the learning features that are going to be displayed. Parents suggested that there should be a mounted simplified



operation manual and project description onto the wall. The everyday person would not know the engineering that goes on behind the scenes. Prospective students suggested that other previous ECE MQP's be displayed. Overall, both parents and prospective students enjoyed the system. It was in general agreement that the system provided an aesthetically pleasing and engaging.

#### **2.3.4. Professor Responses**

To further develop a feel for what assistance the previous MQP received, the team contacted professors and faculty who had an input to the system. The team presented demos to the following people to get a feel for their role within the project development last year as well as to get suggestions for system improvements: Professor Orr, Professor McNeill, Professor Massoud, and Bill Appleyard. Professor Massoud was one of the first people who the team presented a demo for. A demo consisted of the team turning on the system before the person of interest arrived to the presentation. Professor Massoud is the head of the ECE department and throughout the meeting provided a lot of feedback and guidance. He questioned the purpose of the panels themselves and why the choice of LED pixel matrices over the use of a standard LCD TV monitor. Bill Appleyard helped answer that the panels themselves are much more than just panels. They embody what the ECE community is about. The ECE community is about creating innovative and technical solutions. By comparing the LCD TV screens located near the front doors of AK, the team saw that the power panels would be a more engaging and interactive feature to the pumpkin lounge, especially with the capacitive touch sensors as a control feature. Professor Massoud also suggested looking into the MIT room occupancy use. MIT's monitors tell users what classes are in progress and include a countdown until the room is free. Adding this feature would be implementing an entire system change throughout Atwater Kent, of which Professor Massoud suggested that the ECE department could provide funding for.

During the meeting with Professor McNeil, the team updated both him and Professor Mazumder on a list of possible system improvements the team has identified.

He guided the team's first MQP presentations slide and added suggestions to help improve the flow of the slides. Meeting with Professor Orr was insightful regarding increasing energy efficiency. Professor Orr can provide connection to WPI's chief engineer. Through this meeting, the team would be able to gain access to the system's building management system. This program will provide a real-time feed on the energy use throughout the building on campus. It is in the team's goal to be able to showcase this data as the data for the power harvested from the solar panels. Meeting with all of these professors helped provide the team with additional tools and knowledge of who to reach out to regarding certain tasks.

#### **2.4. Final Project Decisions**

The team has decided to implement the following onto the children panels separately: Time and Date, Interactive GIF's, WPI Twitter Feed, WPI Facts, Pizza Friday Countdown, and weather. As for the main panel, the main features to be displayed include: Lab space availability or Motion Sensored Game. The rest will be: mini games, AK map, AK building energy usage, AK solar data, and AK History.

Using the responses from the survey results, the team has evaluated each response based on the following factors.

- Creativity - How interesting, creative, attractive, and unconventional the design is
- Interactivity - How likely will this design encourage interactivity
- Feasibility - How likely will the team complete the design by the end of C-term
- Cost - How much resources will be put into the design

Including these factors will determine the priority of each design and what the team plans to focus on for this project. Below is the value analysis of the three designs

Design	Lab Space Availability	Motion Sensored Games	Open Sourced Page
Creativity	3	3	3
Interactivity	3	3	2
Feasibility	1	2	1
Cost	2	3	3
Total	25	28	23

*Table 1. Value Analysis of Design Options*

### **3. System Upgrades and Additions**

#### **3.1. Challenges of Operating the Previous System**

The team wanted to have the current system in working condition before including new additions to the system. It was something that was agreed upon as otherwise, the errors and problems would only likely worsen later during the project work. Thus, issues were identified to work and fix as priority tasks. The problems that were found along are in the following list:

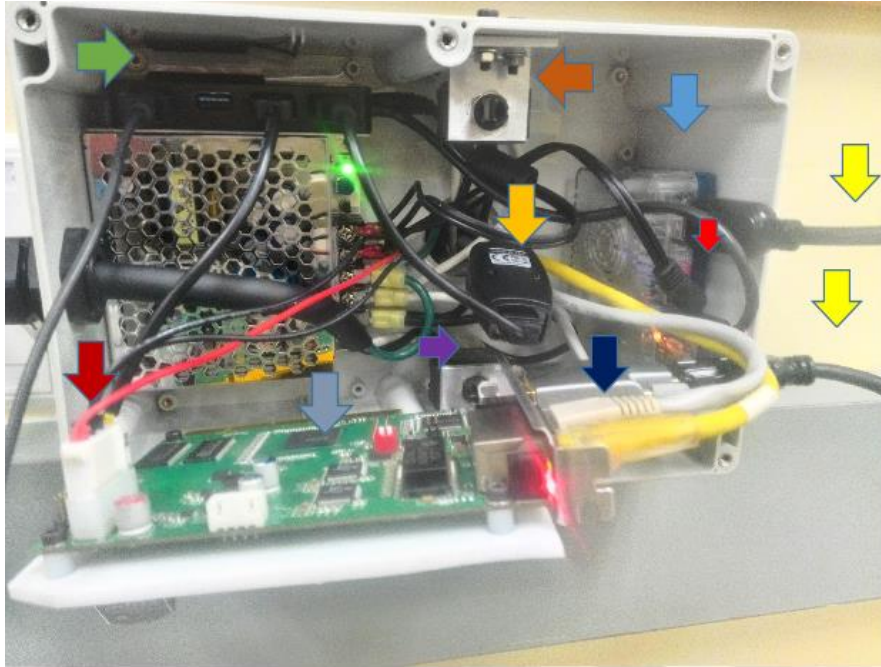
1. Electrical Component Enclosure (ECE) was too cluttered and components were not fixed in place
2. Block diagram was too vague
3. Capacitive Touch Sensor was malfunctioning randomly

During the troubleshooting of these issues, challenges rose that caused roadblocks which delayed the ability to fix them in a planned manner. The challenges were the following:

#### **1. Lack of proper Documentation**

The previous team spent majority of their MQP time building the system and had little time to document their progress. As a result, there are gaps of information within the MQP report. They did their best to include documentation of all things that they deemed important. However, some of the missing information is necessary in order to properly understand the system and to debug error and system failures as they appeared.

#### **2. Electrical Component Enclosure**



*Figure 9. Crowded Electrical Box Enclosure*

The ECE box holds the microcontroller and other components crucial to the power panel.

The major components are color coded and their connections:

- This is the **ODROID** (microcontroller) and its connections:
  - An HDMI,
  - Ethernet cable for internet
  - DC jack
  - And a connection to the **POWERED USB HUB**
- This is the **POWERED USB HUB** and its connections:
  - Connected to the **ODROID**
  - Connected to the **ETHERNET-USB EXTENDER**
  - XBEE module via micro usb cable
- This is the **MOLEX CONNECTOR** and its connections:
  - Directly connected to the power supply via the screw terminals
- This is the **TOP SWITCH**, it is used to turn on the LED Panels in the pumpkin lounge

- This is the **BOTTOM SWITCH**, it is used to turn on the **ODROID**, the **POWERED USB HUB**, as well as the **SENDING CARD**
- This is the **DC JACK INPUT** to the **ODROID**, it is connected directly to the Power Supply.
- These are the two **ETHERNET CABLES**.
  - Please make sure that the **YELLOW** Cable is attached to the U-Port and the **WHITE** Cable is attached to the D Port

This is the **HDMI** Cable, it is used to connect the **ODROID** to the **SENDING CARD** while using a VGA Adapter.

The ECE box is visibly overcrowded with its many components. It is to the point where opening the box results in the components overflowing out of the box. Additionally, there are six screws used to close the box which does ensure security but at a cost of tedious time to remove the cover. The next step is to plug in the USB for mouse and keyboard and change the VGA/DVI cable with a different HDMI cable for a monitor to operate the ODROID. A more detailed procedure of turning on the system can be found in the appendix. It is however clear that the ECE box needs an upgrade in terms of size and ease of use for a faster and smoother operation.

### 3. Capacitive Touch Sensor

As the team worked to identify problems with the Power Panel system, it was discovered that the system would freeze at certain times and not respond to the capacitive touch sensor panels. This would happen randomly such as starting the system, choosing an option or if the system was operating for a certain period of time. The team started to analyze the possible sources for this problem. The focus was on the capacitive sensor regarding how it operated, its components and feedback system. It required more analysis, meetings and assistance from the previous MQP team before finding a resolution to the issue.

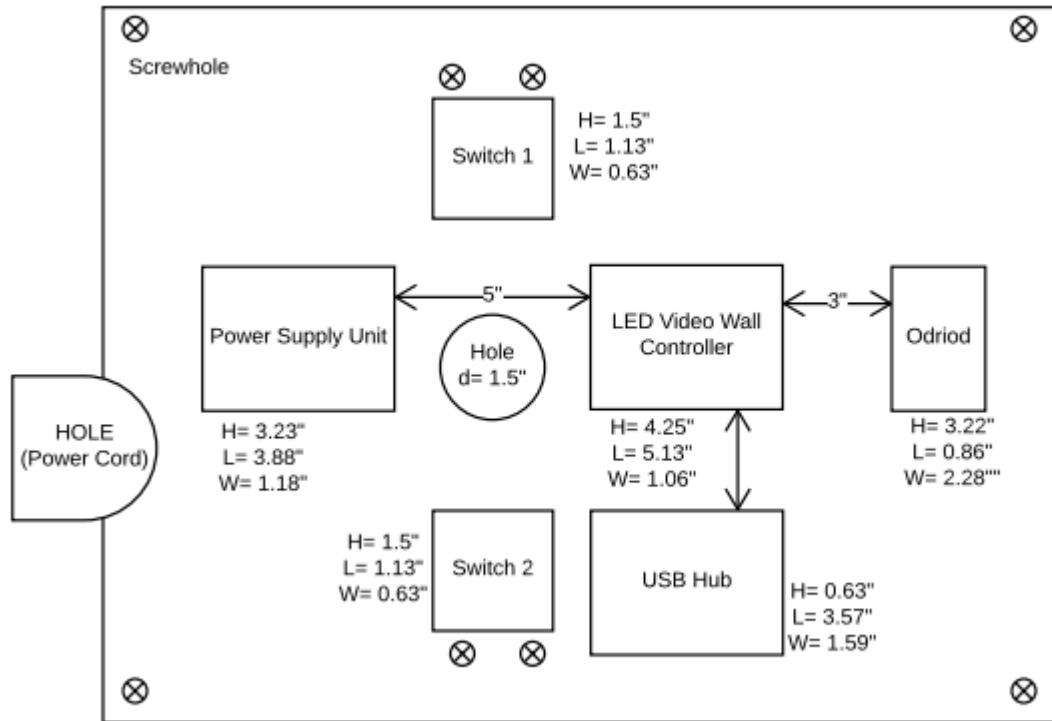
### 3.1.1. Electrical Component Enclosure

The previous MQP team decided to mount the ODROID, LED Video Wall Controller, Xbee Module, Power Supply, and other necessary components into a small enclosure located and mounted on the wall of AK113. The enclosure was a 9"x5"x4" watertight container with a front cover that can be removed by unscrewing the screws. The front cover had a ventilation hole so the inner temperatures of the box do not reach extremities that will result in system malfunctions. The enclosure is significant to the system as a whole because it provides access to power the system on or off. The enclosure also contained a routing hole that allowed wires and cables to be run through to the entire LED display configuration. The enclosure was secure enough for users to not feasibly gain access to the components that run the system.

With these obstacles in mind, the team came up with a solution that would facilitate these problems. To allow for more leeway when ordering a new enclosure, the team contacted AK facilities to provide permission to go ahead and have the bulletin board in AK113 removed since it was not utilized for any purpose. When searching for a new enclosure it was important to have enough space for each component to be able to be mounted on the back panel as well as possibly have extra space for any potential additions to the hardware. The team settled on these circumstances and determined the maximum size for the enclosure is 14"x10"x6". This size was the closest measurement available to what was needed. The team discovered these dimensions based on measuring the components themselves and allowing for some space in between. However, trying to order a premade enclosure runs into an issue to deal with the dimensions the seller can provide. So the team went with an alternative plan for the enclosure.

The team obtained a 10"x10"x6" enclosure that would allow more space for the components to be placed in the box. One of the recommendations from the previous MQP team was to be able to install a small exhaust fan within the enclosure to release hot air that accumulated within the box. With this new box, the team was able to successfully install a new fan. Drilled holes where the fan was mounted allows for efficient passage of hot air to exit the enclosure. There is still quite of bit of clutter but the overall goal is to

still eventually be able to turn the system on/off from the pumpkin lounge rather than having to constantly set it up from AK113. So that the box will not have to be tampered with when trying to operate the system for startup or shutdown.



*Figure 10. ECE Box Component Layout*

The changes made are beneficial because they give each component (ODROID, power supply, LED video wall controller, etc.) enough space and distance so excessive heat won't accumulate to disrupt the system from running. The large enclosure allows for all the components to be securely fastened into the back panel that way no wires, cables, or hardware will be damaged when trying to open the box or turn on the system.



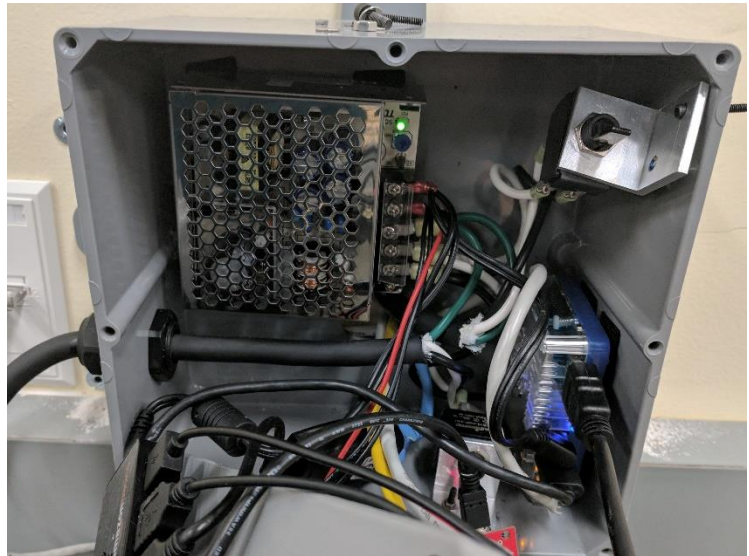


Figure 11. ECE Box

Below shows a table of how the temperature within the enclosure increases over the course of six hours. As shown below the temperature for the old enclosure reaches its peak at around 60 degrees Celsius.

	Current Electrical Component Enclosure	New Electrical Component Enclosure
Duration	Temperature (Celcius)	Temperature (Celcius)
1 hr	35	30
2 hr	46	38
3 hr	54	44
4 hr	61	52
5 hr	62	52
6 hr	61	54

Table 2. ECE Box Temperature data over 6-hour period

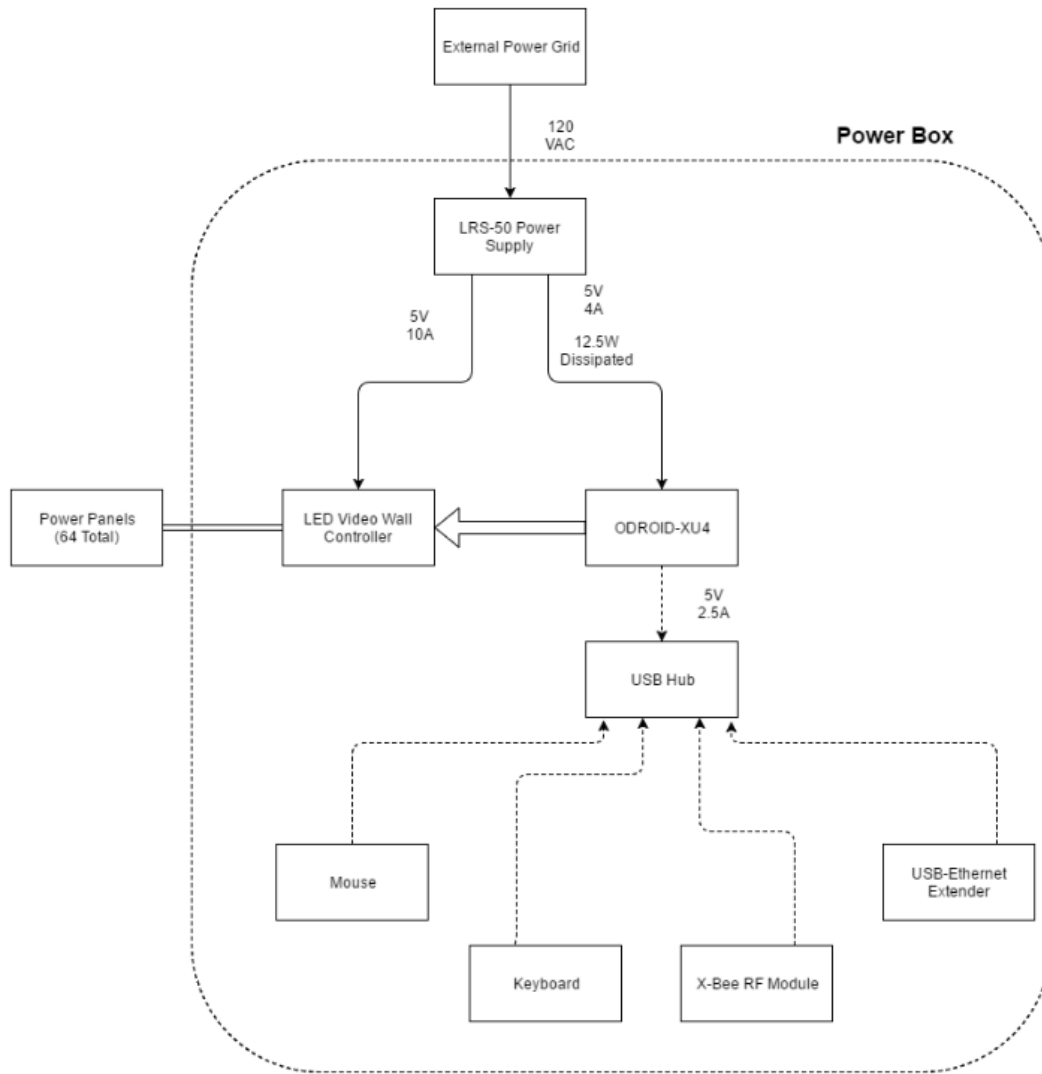


Figure 12. ECE Box Block Diagram

### 3.1.2. Capacitive Touch Sensors

The capacitive touch sensor would stop working randomly when running the system. It was not quite clear as to the reason for this frequent issue from the capacitive touch sensor. To try to better understand the reason, the team reviewed the 2015-2016 report to learn how the sensors worked and the way it was implemented into the system. The sensors were analyzed through its design and found no flaw within it. The way it was designed should have the system running. It was detached from the wall to make sure the

wire connections were not the problem. As expected, the only wire connection was a micro USB wire which joined together from the two capacitive touch sensor and was a direct input to the ODROID with an Ethernet cord.

Not explicit in the report, the team found how the output from the capacitive touch sensor was read by the ODROID to control the main LED display. The capacitive touch sensor basically sends back letters as inputs to control the onscreen display. They are mapped to keyboard letters in 2 3X3 matrix from Q to Y, A to H and Z to N. Once processing is running, the results on screen display can be controlled from the keyboard directly rather than using the capacitive touch sensor. This was something the team did not know and only found out after several tests.

After running the system, the team found that the capacitive touch sensor did not have as many random failures when the grid independent system also turned on. The relationship between the two is most likely a result of the coding and the communication between the two systems. Once the system gained stability, the team focused more in detail as to what caused the capacitive touch sensor to fail.

It was noted that the capacitive touch sensor was fine running the whole system as long as the solar power option was not chosen in the main menu. This option is meant to display the voltage and current input from the solar panels on top of the roof of AK. The information is coming from the Grid Independent MQP which is connected to the solar panel and has voltage and current sensors to read the input. From the previous MQP team's report, it was identified that there are 2 Xbee modules (Receiver and Transmitter) to communicate between the Grid Independent MQP and Power Panel MQP. Other than that, it was not specified how it was programmed for the Grid Independent MQP and its functionality.

Looking specifically at the error when the solar power selection was chosen, it was an error which indicated that the program was attempting to access a location which did not exist. Further research shows that the Xbee modules online provides details on the capabilities and its method of communication. It explains that it used serial port to communicate and needed its own programming. For this part, there was some

programming in the ODROID and some in the Arduino of the “Grid Independent Charging Display” MQP. The ODROID system provided access to read the programming but it did not contain enough commentary to explain the importance of the codes or the program logic. On the Arduino side, the team attempted to access it directly. The Arduino turns out to have a safety feature where once it is programmed, it will keep running the program in binary language and cannot be read in human language. So having access to the source code allows access to study and understand the program logic. Unfortunately, the original source code was lost when speaking to the previous MQP team about it.

Fortunately, one of the members was able to provide any assistance and guidance on the missing information. This provided clarification on the original source code, especially on the communication with the Xbee modules. One of the Arduinos from the Grid Independent MQP was reprogrammed specifically for the Xbee to send the voltage and current information via serial communication.

During a meeting with one of the previous members, the error was displayed while running the system. After the previous team member worked with the programming and some of the coding, he identified the serial port code of the ODROID. He explained that the ODROID had 7 ports which were the different pins available to it. As the ODROID does not have many port connections, the system uses a USB hub to connect the other pins. The Xbee module was connected to the ODROID through one of those USB ports and this was dislocated. The Xbee was tested through different values through the serial port to make the Xbee functional. The test was noting the different ports used until the solar power selection can be accessed. The capacitance touch sensor is now able to function properly with the current settings.

## **3.2. Power Panels Media Upgrade**

### **3.2.1. Children Panels**

The original idea of the LED Panels was to add an art style to it. Some arrangements to the children panel will be able to accomplish that goal as well as make it a welcoming center for tourists and incoming students. The new media will be the “Date and Time,” “Gompei GIF,” and “WPI Fun Facts.” The “Date and Time” was originally on the Main Display, but was best to be placed on a child panel so users of the system can easily see the time and date displayed without having to access it through the capacitive touch sensors. The “Gompei GIF” is an idea to add to the art style of the system where the WPI mascot appears to be making a basketball throw on one child panel and on an adjacent child panel the basketball is made in the hoop. The “WPI Fun Facts” will display a variety of facts about the institution. Just like the scroll of the twitter feed, the fun facts will have a similar style except vertically scroll upwards and display the text.

The team wanted to create a pleasant and enjoyable atmosphere so that the students can use the pumpkin lounge as a relaxation center. Having two panels that appear to be interacting with each other seemed like a way to showcase that.

After some research, the best way to have a GIF in processing is by displaying a sequence of images in loop. For that, each GIF was cropped in squares to match the children panel they will be displayed in and then split up in individual frames that capture the GIF. The frames were downloaded to the ODROID database and loaded into the setup of processing. Then, a counter was created which went up by one every (check) clock cycles until it reaches (check) which represents the total number of frames for the particular GIF. After that, it was a simple matter of calling on the GIF image with counter as input so that the GIF runs with the clock cycle to provide the motion.



*Figure 13. Interactive GIF: Gompei shooting a ball in loop*

The two images shown above are still now but are part of a GIF that will be displayed in 2 of the panels in pumpkin lounge. The left GIF is gompei shooting a basketball without looking at the board while the right GIF is simply a basketball going inside the hoop. The idea is to have the two GIFs running in different panels seamlessly. Once it was understood on how to add GIFs to the panels, the important part will be the timing as both GIF have different lengths. It also has to be a satisfying loop showing the interaction between the two panels.

```
void facts() {  
  if (Count % 500 == 0) {  
    Count_ch2 = Count_ch2 + 1;  
    if (Count_ch2 == Number_of_Fact) {  
      Count_ch2 = 0;  
      fact = 0;  
    }  
  }  
  textFont(font8);  
  textAlign(CENTER); //Center text on coordinates  
  fill(360);  
  
  switch(Count_ch2) {  
    case 0:  
      text("WPI", 670, 45);  
      text("Ranking", 670, 55);  
      text("in", 670, 65);  
      text("US", 670, 75);  
      text("Colleges", 670, 85);  
      break;  
  }  
}
```

*Figure 14. Partial Code for WPI Fun Facts Panel*

The WPI fun facts code is following the same style as the GIF but instead of displaying different frame with cycle, it uses cases to display the facts in the panel. It was

initially attempted to do it exactly as the Gif and took screenshots of the facts but doing so gave a very blurry display in the panels. That is when the switch was made to the case and got clean, easy to read facts on the panel.

### 3.2.2. Main Display Selection Menu

The menu discarded previous selections such as “AK Power,” “Pizza Friday Menu,” and “Time and Date.” The “AK Power” selection was planned to display the measurements of power being used within the building, but the facilities would not allow this information to be displayed so this selection had to be replaced. The other two selections were not going to be used often since there was already a sign that showed the Pizza Friday Menu and the other selection can be accessed from users own phones or watches and clocks. The remaining selections remained on the display and three new ones were added that would encourage users to learn about the Atwater Kent building as well as interact with the system to learn about daily news and information. The purpose of this is to make this user friendly for tourists or other students to utilize on a daily basis.



*Figure 15. LED Panels Main Display Selection Menu*

An issue with the old menu is that it did not clarify the function of the system or how to utilize it properly. The “About” selection was created to ease users into learning the LED Panel system. The capacitive touch sensors do provide a simple display that can be easily understood to navigate the menu, so once the user understands the sensors, they are able to select “About” which explains the rest of the selections and controls.

PUSH UP  
AND DOWN  
TO NAVIGATE  
THROUGH MENU

Figure 16. About Selection Controls Slide

When the user reads this, they will use the up and down arrows to try it out and notice the next two slides below will appear, briefly describing the remaining selections and their purpose. This will give the user the understanding of the menu and the overall purpose of its use.

CAMPUS EVENTS	ANNOUNCEMENTS FOR IMPORTANT	AK MAP	LAYOUT OF AK BUILDING ON ALL FLOORS
GAMES	PLAY GAMES USING TOUCH SENORS	AK MQP	MAJOR QUALIFYING PROJECTS WITHIN ECE DEPARTMENT
SOLAR DATA	DISPLAYS DATA FROM SOLAR PANELS ON ROOF	AK HISTORY	HISTORY OF AK AND ITS SIGNIFICANCE
		AK NEWS	DISPLAY POSTS @ <a href="https://orgsync.com/135668/news_posts">https://orgsync.com/135668/news_posts</a>

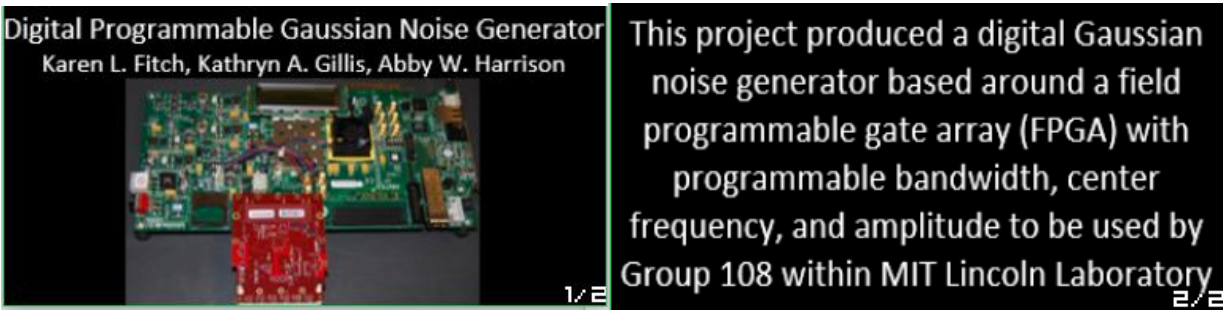
Figure 17. About Selection Description

The last two selections made to the menu are “AK MQP” and “AK History” which fit with half the menu’s theme of focusing on the Atwater Kent building.

GAUSS GENERATOR	FIREFIGHTER GPS
FPGA STOPLIGHT	ROTATION SENSOR

Figure 18. AK MQP Selection Menu





*Figure 19. AK MQP Gauss Generator*

The “AK MQP” selection focuses on displaying some common MQP’s shown in the display cases and in the Electrical Computer Engineering department.



*Figure 20. AK History Slides*

The “AK History” selection focuses on displaying pictures of the building when it first became part of the WPI community and how it came to be.

### 3.2.3. Challenges

Solar Data Glitch (When selecting it on the LED Panel Menu, the entire program would freeze. This can be resolved within a line of code highlighted in Appendix)

### 3.2.3.1. LED Panels Orientation Glitch on twitter and main display

ODROID update 3.10.92-71 was installed on the ODROID and caused an issue with the orientation on the LED Panels. They occur on the main display and the horizontal ticker as shown below.



*Figure 21. LED Main Display and Horizontal Ticker Glitch*

It seems that this is a common issue from other people who own an ODROID and a solution has not been know yet. However, a solution was discovered through the LED Studio Software that is further explained later in this section.

### 3.2.3.2. Children Panel Testing

Updates were made to the ODROID in order to gain access to some remote desktop applications to communicate with the smart mirror. Unfortunately, the update created an issue. It affected the orientation of the processing display on the led panels. Most of them were just moved in terms of location which was restored by changing the coordinates of the media. However, the horizontal ticker and main display panel were showing the glitches as shown in Figure 21. The horizontal ticker had visual glitches

along the tweets which made the text illegible to read while the main display was separated in two different locations.

The team cooperated with one of the previous MQP members. There were different attempts for debugging approach. The first was checking the hardware and wire connections behind the panels, but they appeared to be functional as intended. Another test was to display one of the laptops to the panel display and it displayed the whole screen perfectly onto the LED Panels without any glitches. It was unsure as to why this occurred but it seemed like ODROID update while being connected to the LED Panel system was the only link of creating the distorted display on the media. It was assumed that the ODROID had an issue with the hardware or software so it was best to replace it with another assuming the issue was unrepairable.

As for the glitches, there seemed to be a connection with the way the LED Panels were set with certain resolution and X/Y coordinates. The ODROID display to the desktop monitor was perfectly fine which meant that the problem was not from the code in processing. It also meant that the problem occurred after its sketch output is generated which only leaves the transition from ODROID to the panels. The LED panels have a receiving card installed for this purpose. The sending card is connected with the ODROID with a hdmi-DVI connection.



*Figure 22. Sending and Receiving card for LED Panels*

The problem started after the firmware update of the ODROID. The receiving card which is connected with the LED panel matrix is programmed with an “LED Studio Software Configuration”. The team developed a theory on how this problem occurred which was that the update changed the video graphical resolution which would throw off the software that is trying to display using the old resolution. It was assumed that a new ODROID would fix the problem but after attempting to do so, the display was the same with the glitches. It was attempted to display some laptops and a raspberry pi on the panels. During the testing, it was noted that the pi also had similar issues, one of the laptops couldn’t get some of the panels to display anything at all but one laptop was able to completely display the panels without any issues.

It was still unclear to note the cause since both laptops run on windows 10 and even when changing the settings on the laptop to match the ODROID there was no effect. However, the one laptop which was able to display the panels properly proves that panels and the hardware itself was fine and it was a software issue while sending the data from

ODROID to the panels. With that in mind, the following list was created to solve this issue:

1. Rollback the ODROID software to the previous version – It was assumed that resetting the firmware would reset the panels into displaying the correct coordinates.
2. Using just the laptop to display the panels – It was a backup plan if there was no improvement on the LED display to serve as a temporary placeholder in order to properly demonstrate the features of the LED panels. It would be impossible to implement the idea of using the smart mirror to control the panels as it was meant for the ODROID and not the laptop. Thus, this would have been a temporary solution at best.
3. Use the LED Studio software to reset dimensions for the panels – This is one of the most viable options to utilize as it has the highest likelihood of being able to solve the issue. The problem is that the Software is part of Adrafruit's personal project for which they provide limited support. The only information they have is their setup instruction for their specific project. Adding to that fact, the previous MQP team emphasized the difficulty of the software and stated to not make any changes to it. As it currently works for the most part on the ODROID and fully on the laptop, it is a risk towards the project if changes were to be implemented to the software and the issues become worse.

It was decided to go with option 1 first because it would be the easiest fix if there was such an option available. Unfortunately, it provides no solution. The team reached out to other students who worked with a similar system, researched online in forums from those experiencing similar dilemmas. The only way to do a software rollback for the ODROID running the Ubuntu operating system was by accessing the grub menu. The problem with that was the access to grub depends on pressing 'shift' during boot time. This method did not work nor did other buttons such as 'esc' and 'space.' The team also attempted to directly access the grub menu from the terminal but that was also unsuccessful. It almost seemed as if

the grub menu was non-existent in the ODROID going through the different files, terminal and settings. Researching online for assistance did not provide any information useful to troubleshooting the issue since the community only had experience to solve the shift during boot or using terminal to make changes to the grub menu.

Option 2 was next but after much consideration it was decided the work was not worth the temporary solution unless nothing else worked. As the laptop was able to display properly, a possibility was to use the laptop to display the panels and record a video of the system running in order to show the system working under operation. Although the laptop displayed properly, all of the panels were misaligned with their displays and needed a review on the coordinates. There was also the problem of not being able to access some selections on the main display such as the solar data and the twitter feed (It lacked a specific account approved by Twitter with the proper IP address). The team decided that the time needed to fully shift everything to the correct position for the laptop would be time wasted on a temporary fix if there is the possibility of fixing the display completely.

So, option 3 was the last attempt. The goal was to be able to return the settings to their current settings so that option 2 becomes more viable if the attempt does not work. Screenshots of the settings were taken as a trail to redo any changes. The screenshots are included in the appendix as well as a description on how to operate the LED Studio configuration software. The team worked with the software and tried to make minor changes to the settings and better understand it. Eventually, it was identified that changing the resolution was able to minimize most of the glitches. This finding, and the error following the ODROID update, coincide to express the ODROID's old firmware had a certain resolution which was matched by the sending card which controlled the LED Panels. The firmware update for the ODROID made some changes to that resolution which no longer matched the sending card and thus created the glitch in response. Changing the sending card resolution with the LED Studio configuration software brought it to a resolution close to the new updated firmware's resolution and eliminated

most of the glitches. The minor glitches left are because the configuration software only has custom resolution so it cannot be customized to match the exact resolution.

### **3.2.3.3. Unexplained ODROID Malfunction**

After fixing the glitches the LED Panels and demonstrated it working properly, another issue was presented. Without adding further changes to the working ODROID and panels, the LED panels connected to the ODROID did not display any media. The ODROID was assumed to have an internally damaged component so another ODROID was used to replace it. It was initially claimed that the issue came from the LED Software, then the HDMI cable, or even with the components inside the ECE box. However, tests were run to check what components were damaged by displaying the images from a different processor (laptop, raspberry pi) which were capable of providing the display. The ODROID did not and the likely possibility of this was it being damaged or having the wrong HDMI cable. When replaced with the new ODROID, the display appeared again. This fixed the problem but after demonstrating it on Project Presentation Day where the system was working perfectly, the new ODROID also had the same malfunction where it did not display anything in the panels even though other devices could. It hints at an unexplored problem with the ODROID.

## **3.3. Solar Grid Independent Charging Station**

### **3.3.1. Current System and Goal**

With the idea of creating a user friendly console to be provided to the public of Atwater Kent, the Solar Charging Station needed to be assessed for its functionality and ability to be relocated to the first floor from the third floor of the building. In order to make this a feasible task, there were three big factors that must be clear to pursue and accomplish: (1) The charging system must be functional when relocated a further distance away from the third floor to the first floor, (2) it must be enclosed in a secure area that cannot be tampered with easily yet present the components that make up the system in a clear way, (3) and the system should still hold its purpose of providing the

functionality of a charging station as well as provide information of the solar panel's live data feed.

The functionality is priority to ensure the system will perform when relocated, so it was best to run the system in its current state in the lab on the third floor. This testing of the system is shown below in Figure 23.



*Figure 23. Solar Charging Station with Xbee Module*

What should be noted about the system is the removal of the LCD screen on the top left of the picture and is replaced with an Xbee Module. This was a decision made as a response to the recommendation of creating a larger display for the solar panel's data feed through a wireless, zigbee communication from the Charging Station to the ODROID, which presents the data onto the LED Panel menu as a selection.

It is noted that the LCD screens display 0.000W and this was a known issue the system produced before the installation of the Xbee Module. Overtime the system did lose the LCD monitor functionality to properly state the power being distributed, but did



provide the opportunity to apply a better usage of presenting the energy usage of the Solar Charging Station that can provide the awareness of Solar Panels providing power at different times of day for the public to use.

### 3.3.2. Procedure

Knowing that the Charging Station is functional and able to perform, the next task was to make the system presentable for use by positioning it next to the LED Panels to show the connection between the system and its live solar data feed. This means the system should be visible to the public but should be enclosed in a secured manner. Thus leading to the usage of Acrylic plexiglass, specifically 1/4 inch thick. Figure 24 shows the plan of the casing for the components of the system as well as measurements made for each of the sides of the case. Figure 25 shows the bottom of the case with the components laid out. What should be noted about the measurements is that there will be six different slabs of plexiglass to make up to box-shaped enclosure. The measurements were taken based on the sizes and shapes of the components that make up the system and the distances between each component.

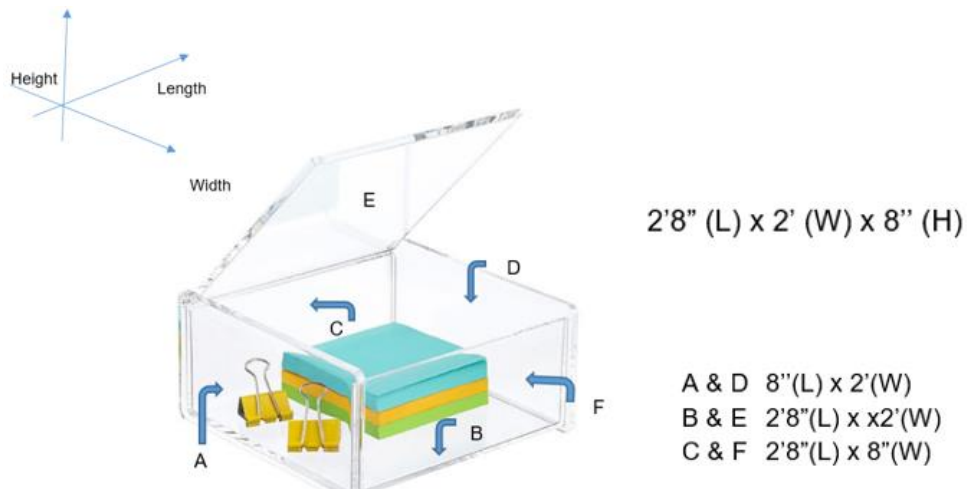
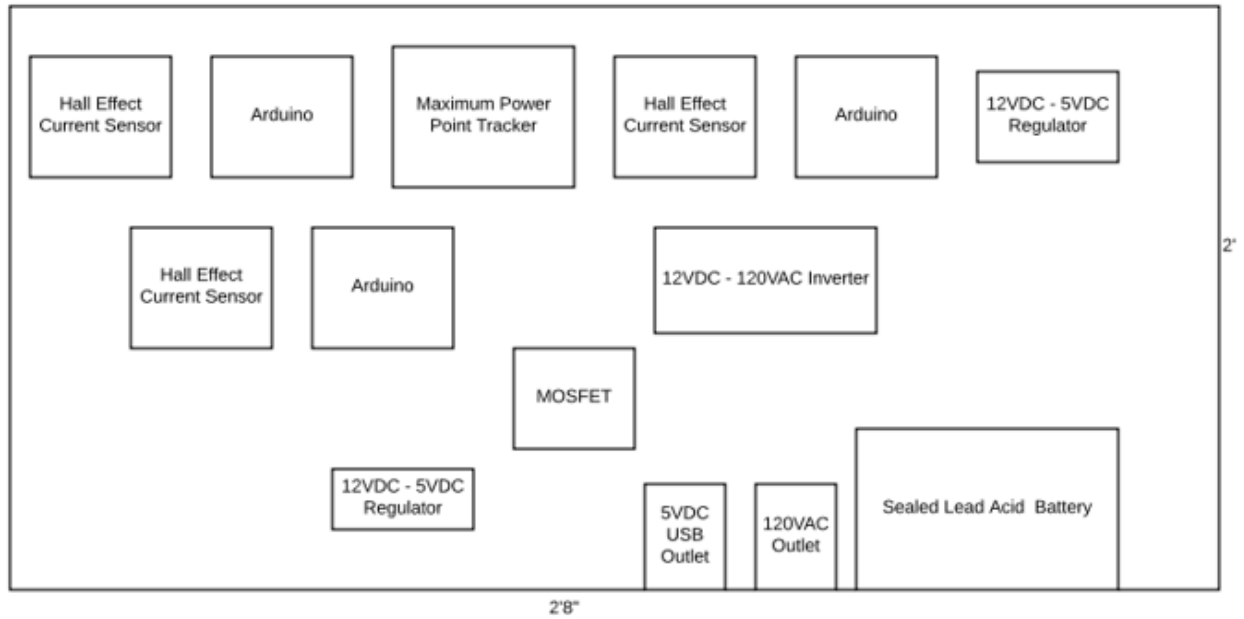
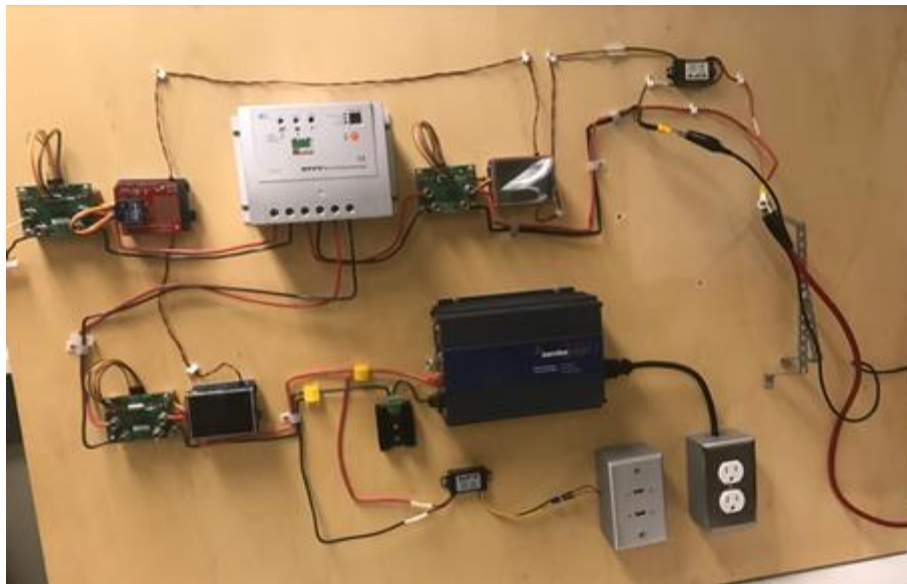


Figure 24. Acrylic Case with Measurements



*Figure 25. Acrylic Slab with Positions of each Component*

Next was removing the system from the wooden board it was attached to within the third floor lab. The system attached to the board is shown in Figure 26.



*Figure 26. Solar Charging Station attached to wooden board*

It was secured with numerous nuts and bolts that included time to remove the components as well as note the wire connections each one had. Once the components

were removed from the wooden board, it was laid across on an extra acrylic slab that was used to provide an idea of the system taking up the size that was noted in Figure 24. As seen in Figure 27, the system does have an issue of wires being too long to fit within the size limit, which does request it to be rewired to adjust the direction as well as shorten the length when connected between two components.



*Figure 27. Solar Charging Station removed from the wooden board*

With the layout of the system planned into an acrylic case enclosure, the next step was to also extend the connection of the solar panel to the system with a longer cable. Below shows Figure 28 which is the cable that was used to extend the cable connection and is the type of wire gauge (14-2) compatible with the solar panel. With the assistance of the head of facilities of Atwater Kent, the cable is able to extend from the third floor down to the first floor where it is located in the AK113 lab. It was then ensured that the new cable is spliced together with the old one and placed near the ceiling.



Figure 28. 14-2 Wire Gauge Solar Panel Cable

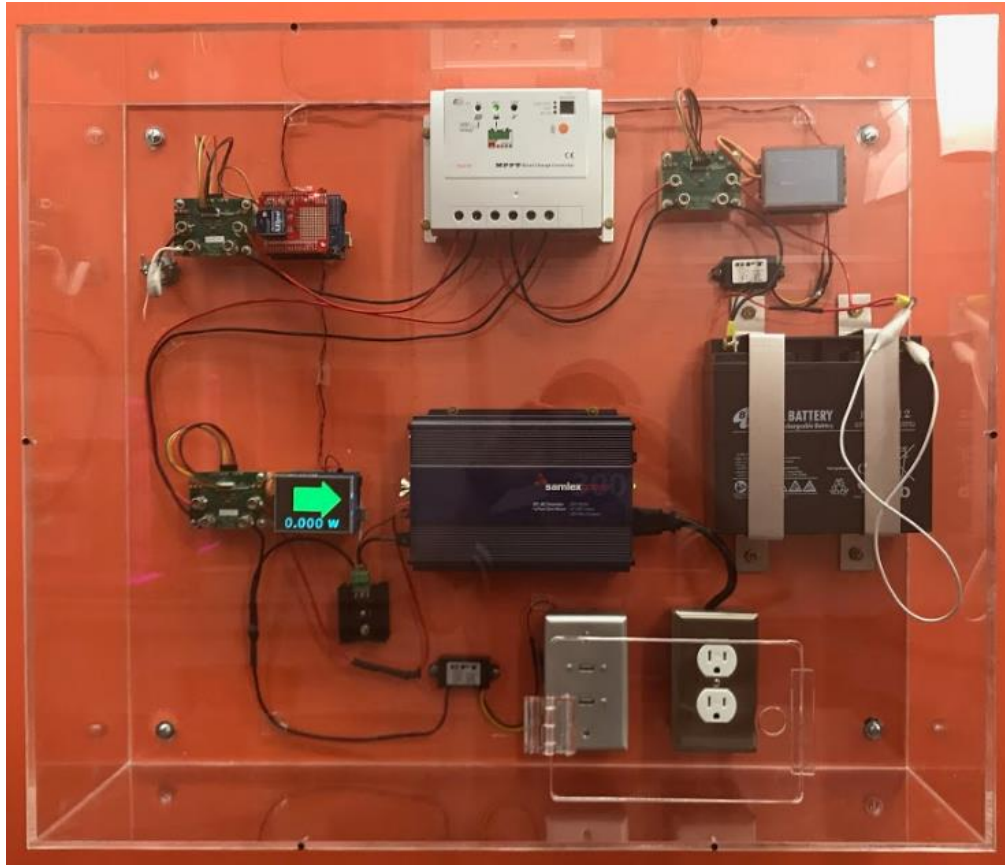
With the ability to relocate the system to the first floor and planned out the positions of the components within the acrylic casing, the final workings are to create the casing, rewire the components, and place the system on the wall. The acrylic material does have a ¼ inch thickness to ensure the case is stable enough to hold the components and the six slabs of acrylic are able to hold together when glued. With the assistance of the electronics technician, the acrylic slabs were cut to their planned sizes. Next was placing the components onto the bottom of the case, where the plastic covering was used to mark the locations of the components and their areas of placing screws to attach to the acrylic slab. This provided a safe excuse of making mistakes when placing the components as well as change location if needed. With the components placed, new wires were used provided by the electronics technician which are 18 gauge red and black wires. These required use of wire splitting tools and soldering techniques for some components, but the goal was to make the system have an aesthetically appealing display to users of the system. Once this was complete, the plastic was peeled off the slab that was on the cover that holds the components since the case was ready to enclose the system. The slabs were glued one by one to the slab holding the components and settled within five

minutes of gluing. The last slab that is used to cover the top was placed using six different screws in order to allow the option to reopen the case if there is an error with the components or provide features to the system. However, this did become an issue to allow access to users who wish to utilize the charging system, so a small door hinge was created in front of the power outlets for users to access the system with ease.

### **3.3.3. Finalization of System**

The last step was to place the cased system onto the wall. After speaking to the head of the Atwater Kent facilities, the team was given the permission to place it next to the main display of the LED panels. It should be noted that this was allow to hang on the wall since there were not metal pieces inside the wall that the system can interfere with, so it was safe to place it along with four different bolts to hold the case. Once the system was installed, the 14-2 gauge cable connected to the system and the system became functional to provide charge. The final system can be viewed in Figure 29. which is now capable of sending the solar data feed from the solar panel located on the roof down to the first floor of the pumpkin lounge and provide an option for users to charge their devices.





*Figure 29. Final Installment of the Solar Charging Station in the Enclosure*

### **3.4 MQP addition Criteria**

1. Interactive – Students/Visitors are encouraged to use the system
2. Comfort - System is located in Pumpkin Lounge so the system’s interaction features should not infringe upon any user’s comfort
3. Distraction free – Pumpkin Lounge is at all time of the day used as a study area. New addition to the system cannot be obnoxious or distracting to the students’ education
4. Ease of Use – New addition should help using the system easier
5. Aesthetically pleasing - The new addition will be visible in the main lounge area of the ECE department and will be used by prospective students.

#### MQP addition options:

1. Interactive Game – The original idea of our MQP team was to create an interactive game that both visitors and students can play to learn about the system. It was hoped it would be the initial attraction that would have people use the system for fun and then they would learn and use the rest of the system features. However, while it would be interactive, it could be a source of distraction for other students in the lounge who are focusing on their school work.

2. Music Speakers – This was an idea explored where it would help students relax by making the pumpkin lounge an open music room. It would be an interesting aspect that is not seen in any WPI buildings and can definitely be a tool for relaxation and comfort in the lounge. It is however hard to pick a genre of music that everyone can agree with and as Pumpkin lounge is the heart of AK, a lot of people will be influenced by the sound. While some students might enjoy the addition, it can definitely be a source of distraction for others as there is no easy way to isolate the music to a small location within the lounge.

3. Smart Mirror – This is the idea the team came up with when exploring the criteria. It would be user interface which can help control the system right from the pumpkin lounge without infringing on anyone’s comfort or being a source of distraction. It is interactive by design and matches the criteria for the addition.

It was then decided to go with the Smart mirror user interface as a new addition to the Power Panel system. As it gives an option to control the system from the pumpkin lounge, it was decided to move the Grid Independent MQP there from AK 317B as well. The NECAMSID lab is off limits except to the Professor and MQP teams working there so normal visitors and students cannot access the lab room. Bringing down the Grid Independent MQP which sends the solar data to the Odroid has the added benefit of actually utilizing the solar power. It has an inverter which powers a wall outlet that can be used to charge small devices like cell phones and laptops.

### 3.5 Smart Mirror

#### 3.5.1. Block Diagram and Component Analysis

With the new feature being the smart mirror, some investigation was required to understand what components are necessary to extend the system's interactions. The main components for the mirror will be the computer monitor, the sheet for the mirror, the Raspberry Pi 3 microcontroller, and the Gesture Sensor. Below is the Block Diagram of the 'Smart Mirror.'

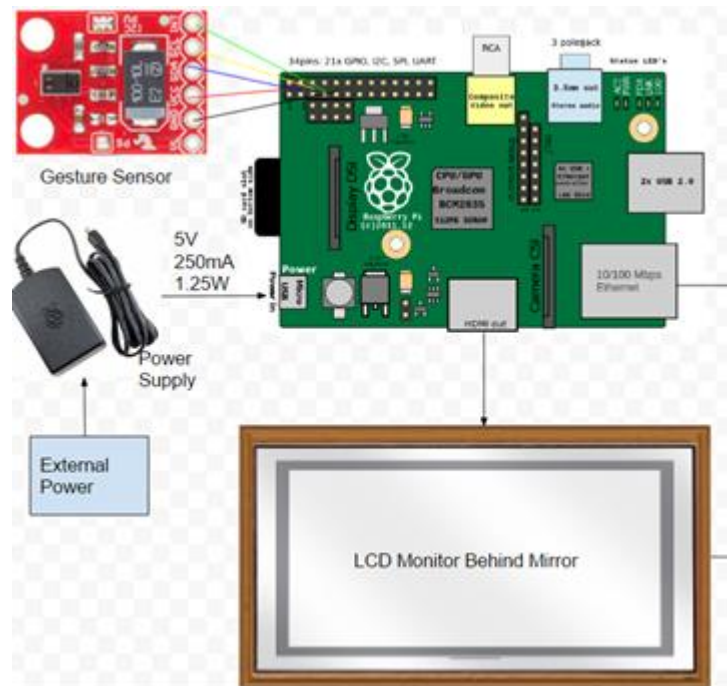


Figure 30. Smart Mirror Block Diagram



The components that make up the ‘Smart Mirror’ are listed below.

- Two-Way Acrylic Sheet \$86.67
- 2 by 4 wood \$10.50
- LCD Monitor \$169.99
- Raspberry Pi 3 Model B \$39.99
- HDMI Cable \$5
- Power Supply (With Raspberry Pi)
- Gesture Sensor RGB \$14.95

The ‘Smart Mirror’ uses the Raspberry Pi 3 microcontroller that is connected to the LCD Monitor through an HDMI cable in order to display the image through the mirror and the Ethernet cable is able to communicate the data between the LCD Monitor and the microcontroller. The data comes from the RGB Gesture Sensor where users will use their hand movements to interact with the ‘Smart Mirror.’ The entire ‘Smart Mirror’ system will have a power supply that provides 5V, 250mA, and 1.25W to the system through a micro USB cord. The power supply will be connected to the external power of the AK building.

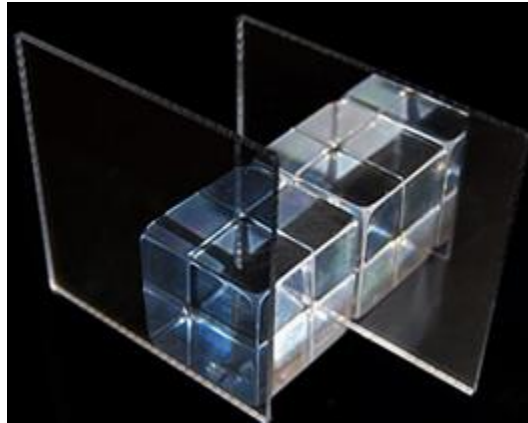
The size of the mirror is dependent on the monitor, so the monitor must be considered first. With that in mind, the LCD Monitor below has been chosen.



*Figure 31. LCD Monitor*

The Monitor is able to connect to the raspberry pi with ease since it comes with an HDMI slot. It is a 27' (25.3' x 18.3') display which is an appropriate size for users to clearly see the screen through the mirror.

When researching how the smart mirrors are configured, the common material used for the sheets are two-way acrylic sheets. These sheets provide a reflection on one side and the other side appears transparent. This sheet is able to hide the computer monitor behind the sheet and when the monitor is turned off the sheet will appear as a mirror.



*Figure 32. Two-way Acrylic Sheet*

These sheets can be custom ordered and sized in various ways. In order to make the smart mirror, the acrylic sheet must be slightly larger compared to the monitor. Fortunately, the company TAP Plastics offers customization for these sheets and can be ordered in any size that can fit the criteria of the smart mirror.

The smart mirror requires the microcontroller in order to provide the interface. Choosing the raspberry pi 3 is relatively cheap and contains simple instructions on running the operating system onto the monitor and easy set-up. There is also a large community behind the pi 3 that provides feedback and input on setting up the smart mirror configurations.



*Figure 33. Raspberry Pi 3 Model B*

The Gesture Sensor is used to recognize motion detection and output the detection of the motion passing over. The way it will be utilized for the ‘Smart Mirror’ is it will detect when a user moves their hand across the sensor. Doing this will turn on the system and the user can move their hand in a certain direction to move through a menu selection. The user can move their hand towards the sensor and it will recognize that motion and respond by selecting the option on the menu.



*Figure 34. Sparkfun RGB and Gesture Sensor*

Push buttons will be another option for the ‘Smart Mirror’ if there is a need to switch the way users will interact with the ‘Smart Mirror.’ Push buttons are simple to install and can be easily accessed by the user to interact with.

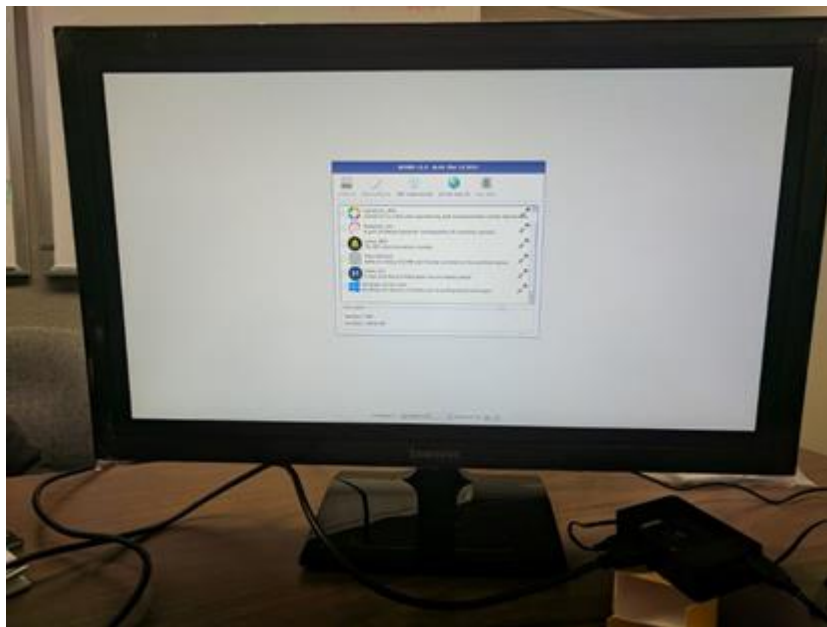


*Figure 35. Push-Button*

### **3.5.2. Installation Process**

The process for setting up the Smart Mirror required a lot of minor tweaks such as the proper operating system installation, being able to connect to WPI's Wireless Internet, configuring the system clock, etc.

#### **3.5.2.1. Installed Raspbian OS**



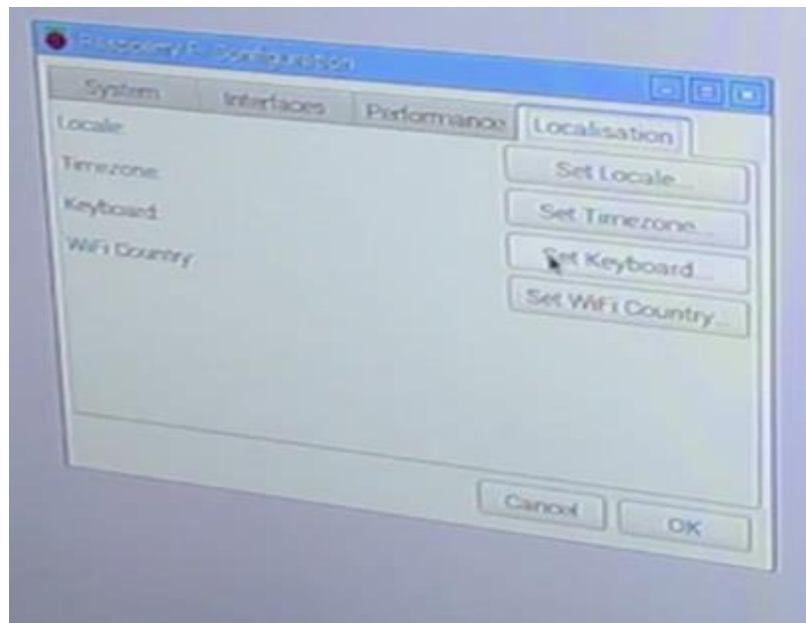
*Figure 36. Initial Setup when first booting up Raspberry Pi*

When first booting up the Raspberry Pi the user is asked what operating system they want to use when using the Raspberry Pi. There are plenty of options available to

choose from but the most basic and feasible option for the purpose of the Smart Mirror was to choose Raspbian. When going through this OS installation process it was required to have internet. This provided an issue registering a Raspberry Pi onto WPI's internet since the setup is complicated. A work around this issue was through the use of a hotspot device. Once connected to the internet, the installation began to download.

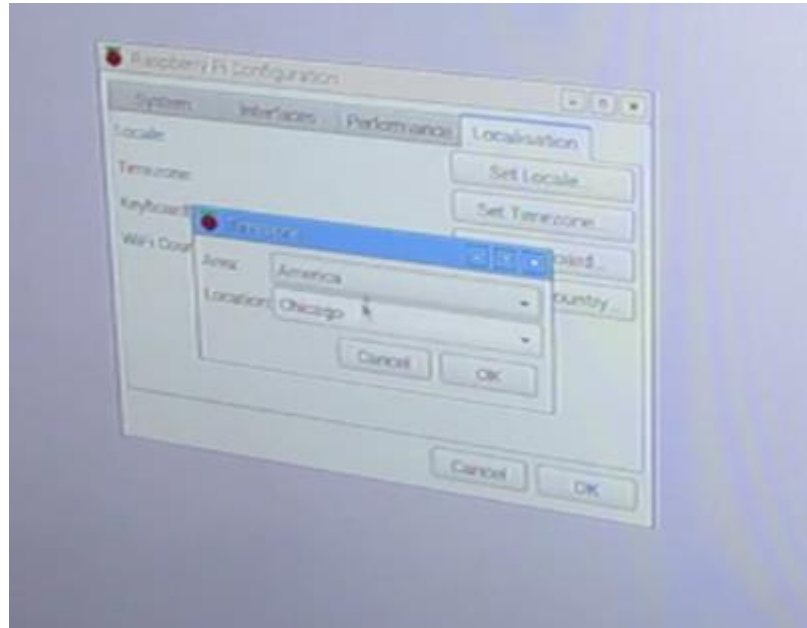
### 3.5.2.2. Configured English (US) Keyboard and System Clock

The Raspberry Pi is defaulted to the British English Keyboard layout which made it slightly more difficult when setting up the Smart Mirror software. So it was required to set up the keyboard to US English.



*Figure 37. Changing the keyboard regional language*

This was simply done by going to the drop down menu at the top left of the desktop screen and selecting Preferences>Raspberry Pi Configuration>Localization and then clicking on Set Keyboard. Then a list of keyboard configurations are laid out to choose from.



*Figure 38. Menu options for selecting Time Zone*

As shown in the figure above the time zone/system clock can be configured from the same menu as the keyboard settings. Then the user is prompted to select their location.

### **3.5.2.3. Setup Wi-Fi access with WPI's Internet**

Setting up Wi-Fi Access for the Raspberry Pi was no simply task. WPI does not have a transparent way of connecting Linux systems to its internet unless Network Manager is installed. When going to the WPI Helpdesk located in the Gordon Library, they informed the team that the Helpdesk does not have knowledge on how to setup a Raspberry Pi on WPI Wi-Fi. When the team first set up the Wi-Fi for the Raspberry Pi, a specific certificates needed to be downloaded from WPI Network that would then be used with `wpa_supplicant`. This information was obtained from an online forum that a past WPI student wrote that contained information about the `wpa_supplicant`. Setting up internet access with `wpa_supplicant` is possible but very tedious and unsafe because it leaves the student's username and personal password in a text document that can be

accessed without proper security. This can be shown in the example wpa\_supplicant file below.

```
1 #ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
2 #update_config=1
3
4 network={
5     ssid="WPI-Wireless"
6     key_mgmt=WPA-EAP
7     proto=WPA2
8     pairwise=CCMP
9     group=CCMP
10    eap=TLS
11
12    identity="[YOUR_WPI_EMAIL]@wpi.edu"
13
14    ca_cert="/home/pi/certs/CA-[A_BUNCH_OF_NUMBERS].pem"
15    client_cert="/home/pi/certs/certificate.pem"
16    private_key="/home/pi/certs/certificate.pl2"
17    private_key_passwd="[YOUR_WPI_EMAIL_PASSWORD]"
18
19    priority=1
20 }
```

*Figure 39. Format of wpa\_supplicant, proper certificates and personal password*

It is suggested that if the Raspberry Pi loses its WPI Wi-Fi access, it should be taken directly to Network Operations located in Morgan Hall. They were able to efficiently setup the Raspberry Pi onto the WPI Wireless Internet using Network Manager. The reason why Network Manager is less tedious to use is because it allows users to choose what certificates are needed instead of having to input the file path to each certificate.

#### **3.5.2.4. Installing Smart Mirror Software**

Before installing the necessary packages for the Magic Mirror software, it is advised that space is freed from the SD card. Once all the extra software that is preinstalled is removed, it can then begin installation with internet access. The team followed this very detailed step-by-step forum that shows the command used to install the packages for the Magic Mirror software. It also shows how to reorient the screen position

to either be in portrait or landscape mode. Using this software gave a base for what the team wanted to display onto the Smart Mirror.

### **3.5.2.5.      **Obtained API Key Codes to Obtain Live Weather Feed****

After installing the software and running it, the team noticed that the weather forecast wasn't displaying the correct information. This is because the software grabs its information from a website called OpenWeatherApp. An account is needed to obtain an API code to be entered. A location ID is also needed to display the weather forecast from Worcester, MA which can be obtained from a list[2] of cities located on the OpenWeatherApp.

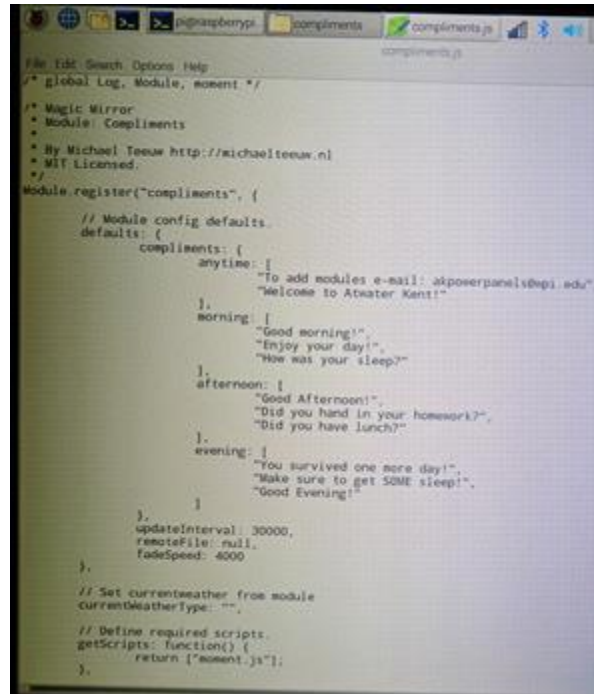
### **3.5.2.6.      **Implemented Automatic Startup for Smart Mirror Software****

The team wanted to add the feature that the software being used for this project would automatically initiate whenever the Raspberry Pi was booting up. There is another forum[3] that has detailed description of what command will start an installation of a production process manager for Node.js applications. This allows to keep certain applications alive forever, reload them, and aid in common system admin tasks.

### **3.5.2.7.      **Customized Compliments Relative to Atwater Kent Users****

Within the software folders of the Smart Mirror there are various modules that are used to display information onto the LCD screen. Specifically looking at the compliments folder will provide a text file that can be edited to display whatever compliments/phrases that were relative to users of Atwater Kent





```
File Edit Search Options Help
/* global Log, Module, moment */

/* Magic Mirror
 * Module: Compliments
 * By Michael Teeuw http://michaeltteuw.nl
 * MIT Licensed.
 */
Module.register("compliments", {
  // Module config defaults.
  defaults: {
    compliments: {
      anytime: [
        "To add modules e-mail: akposerpanels@wpi.edu",
        "Welcome to Atwater Kent!"
      ],
      morning: [
        "Good morning!",
        "Enjoy your day!",
        "How was your sleep?"
      ],
      afternoon: [
        "Good Afternoon!",
        "Did you hand in your homework?",
        "Did you have lunch?"
      ],
      evening: [
        "You survived one more day!",
        "Make sure to get SOME sleep!",
        "Good Evening!"
      ]
    },
    updateInterval: 30000,
    remoteFile: null,
    fadeSpeed: 4000
  },
  // Set currentweather from module
  currentWeatherType: "",
  // Define required scripts.
  getScripts: function() {
    return ["moment.js"];
  }
});
```

Figure 40. Example config file of compliments displayed on Smart Mirror

As shown in the image above, it is simple to configure what compliments can appear as well as implementing other within the Smart Mirror.

### 3.5.3. Smart Mirror Features

The purpose of the Smart Mirror is to allow users of AK to control the power consumptions settings of the LED Display through brightness, power on/off, or sleep mode. It also contains all the default modules for users of AK to access readily available information such as weather forecast, upcoming holidays, current events, etc. An additional module aside from the default ones that was added is being able to display WPI's Instagram photos onto the Smart Mirror. An online tutorial (<https://github.com/kapsolas/MMM-Instagram>) was used to assist in creating the module to be compatible with the Smart Mirror software. This method did require another API code to be linked with a current Instagram account. (<http://jelled.com/instagram/access-token>) With this website generating API codes it allowed the team to display pictures of WPI's campus and students on the Smart Mirror.

## **4. Future Recommendations**

The following section will be divided into three parts to understand how to improve each individual system component in order to reach the goals of this MQP.

### **4.1. LED Panels**

The LED Panels themselves are in a functional state with the panels themselves being tested for long periods of time. However, issues that the team has not been able to fix include being able to start the panels themselves remotely. The necessity for this stems from not being able to work on both the aesthetics of the power panels themselves and testing out what changes made look like since there needs to be a manual change between the HDMI or VGA port to and from the programming card. There were two attempts made to address this issue.

The first is to enable wake on LAN through the Odroid. The Wake on LAN(WoL) functionality serves to be able to turn on computers while it is on low power mode or off. This attempt was unsuccessful because the Odroid does not have a BIOS system in which would be able to enable WoL. The second attempt was attempting to create the electrical switch on the wall of AK113 to face the same direction as the power panels themselves. This way, future students could turn on the panels without having to go to the other side of the wall to unscrew the lock box. A recommendation that the team considered but did not attempt was removing the use of the Odroid and replacing it with a Raspberry Pi. The reason for this being that there are previous examples online on how to remote desktop into a Raspberry Pi.

#### **4.1.1. Electrical Component Enclosure**

One of the main concerns when the team first learned about the MQP was that of heating issue. This issue was looked into and noted it was not a main concern. What is concerning is how cramped the components in the ECE box is. The team was able to receive a size close to the one that was determined to be the maximum for the enclosure, but it did not provide a large enough space for the components. Since it is not a major

concern, the team installed everything and added in a cooling fan. There were holes drilled onto the front of the enclosure to allow hot air to come from out of the box. This box can be changed for aesthetic purposes but it should be noted that there has not been a heating issue detected.

#### **4.1.2. Games**

Within the Odroid's Processing code, there are two games that are featured. However, while troubleshooting the system in AK113 many students from the AK community have presented their curiosity for a method of submitting games to display them on the LED Panels. A way to approach this would be to allow students to meet with any future teams working on this project in order to understand how Processing works on the Odroid. It is noted that some robotics and computer science students have used this program before to design and display media. These games should be approved by future teams before uploading the program into Processing to be displayed.

#### **4.1.3. Solar Panels Installation**

The team researched the feasibility of installing the remaining solar panels from the 2011 MQP entitled "Renewable Energy Applications." After making the decision to construct a solar energy system, discussion of system design and location for the panels began. Through Professor Looft, contact was made with WPI alum Jim Dunn, who has a solar panel installation company called Future Solar LCC. The 2011 team discussed with Jim Dunn about the possibility of installing solar panels onto the roof of Atwater Kent. Mr. Dunn mentored the team and donated six solar panels of three different models so the team could compare the specifications of each model. However, the team only managed to install the system was designed to handle in excess of 230W, 45V, and 9A to take input from all three different types of panels. Upon reviewing the specification sheets for the remaining solar panels, it was expected that they would not fulfill the power requirement of powering the power panels. In an attempt to resolve this issue, the team looked into the possibility of buying new solar panels to install that could power the

power panels themselves. This opened a conversation with facilities and the Atwater Kent Electrical Engineer Manager, Professor O'Rourke.

The take away from this meeting include not being able to access the roof after it collapsed in 2016. The reason for this being that the solution after the roof collapsed was to apply a rubber layer which acts like a large tarp on top of the roof. This also means that the solar panels cannot be attached to the roof or else they would cause holes. An effect coming from this is that none of the current solar panels are attached to the roof. This means that they are no longer in their most optimal position on the roof. They are currently held down through the use of cinder blocks. This poses a challenge to add more solar panels because they too could require cinder blocks which would increase the weight on the roof. This creates a detrimental situation and can cause the roof to collapse.

## **4.2 Smart Mirror**

### **4.2.1. Remote Desktop**

In the section above entitled remote desktop on the Odroid, it was made clear about the different approaches taken to complete this. Additionally, it is recommended to switch the Odroid to a Raspberry Pi. An additional benefit to doing this would be that the Magic Mirror interface on a Raspberry Pi itself. A script can be written onto the Magic Mirror modules that turns on the Power Panel's Raspberry Pi.

### **4.2.2. User Control**

It is important to note the team attempted to have users control the Smart Mirror. The team used a module found within the Magic Mirror websites which used the Raspberry Pi's IP address and SSH settings to be able to navigate through the Smart Mirror through a website. This website would be displayed on the Smart Mirror itself and was meant for users to be able to select what they wanted to see featured on the screen. After mounting the panels, the team left the power panels on for an hour to note the reactions of the addition of a Smart Mirror. Within hours, the AK Community had used the website to modify the interface entirely. In conclusion, it was not intended to grant

this much access to the AK community. Instead, the team suggested limited accessibility through the use of buttons or gesture control. Both of these features were purchased and can be found in the NECASMID Lab in Atwater Kent.

The SparkFun RGB and Gesture Sensor can be installed on the front of the magic mirror so nothing is blocking its ability to sense the user's hand in front of it. The gesture control works by being able to "sense" the direction of movement from the user waving their hand in front of it. This provides feasibility of moving from modules left and right or up and down if the modules are displayed in a console setting, which is a module found on the Magic Mirror Modules. The use of buttons can do the same purpose of navigation, however, there is a higher risk of the buttons malfunctioning eventually. This can come from everyday wear and will need additional and frequent troubleshooting even after the MQP team is done.

#### **4.2.3. Lab Occupancy**

A feature that was highly sought after was having a way to measure free space in Atwater Kent. This is a feature that would look better on the Smart Mirror because the high amount of information would appear pixelated on the power panels. The team approached Professor O'Rourke on how to approach this problem but due to time constraints it was not a priority focus for this project. The first consideration was being able to have weight sensors placed on the chairs of the lab spaces of Atwater Kent. The weight on a chair would indicate that the lab seat was being used. On the Smart Mirror, there would be a module that included all the lab spaces in Atwater Kent and displayed which seats were occupied. A different approach to this addition would be to have occupancy sensors which measures the amount of people that enter and leave the labs. This is a problem on its own because it has to incorporate permissions from WPI facilities. A third approach would be to monitor the user logins within the desktops of the labs. This approach is risky because it is considered private information if someone logs onto the computer.

## 5. Conclusion

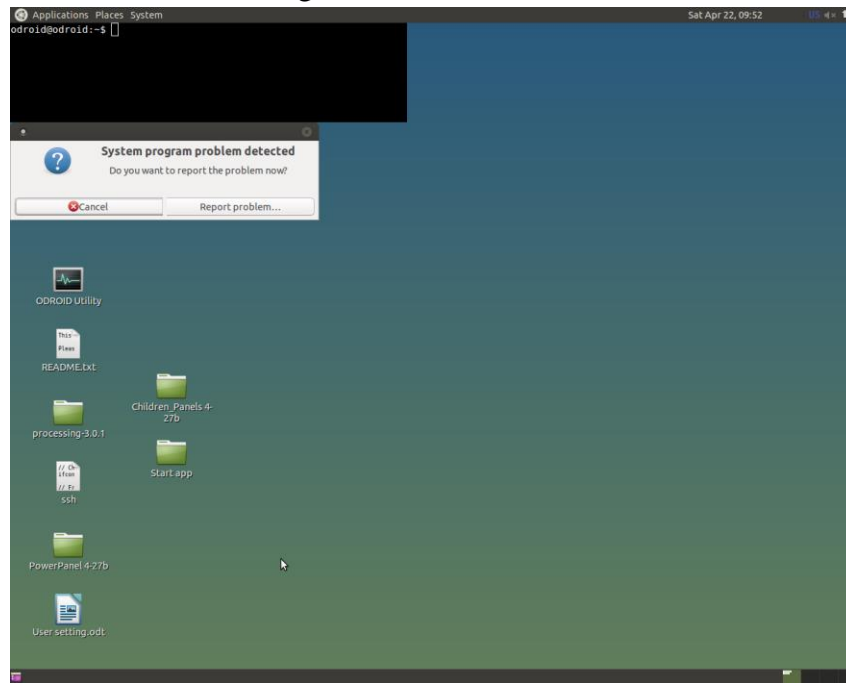
To summarize, the team can make note of the following items. The team identified four focuses in A-term: improving documentation of the system and the system's block diagrams, enabling the system's capacitive touch sensors, finding a method of having the system switch from on the grid to off the grid energy, and changing the system startup process. With B-term, the team has finalized improvements of the previous MQP's system block diagrams with the help of members from the previous MQP team. The team has also added in the block diagram of the proposed improvement of a magic or smart mirror feature to the existing diagram of the entirety of the system, as well as added in a block diagram of the Smart Mirror system itself. The previous system's capacitive touch sensors have been fixed through software troubleshooting and now respond to human touch and can select the options in the system's digital menu. There were physical complications to having the previous system run through the use of the solar panels on the roof. However, a block diagram was proposed to help power the current system using the current panels found within AK. There was also another solar panel system proposed that included panels which could be brought. This was done to provide a future recommendation of system improvement to using less energy. Another way the team tackled both issues of high energy consumption and to address the complications of system start-up was to propose the idea of implementing a smart mirror. This would help address the energy consumption issue because the Smart Mirror would allow the overall main display to be turned on/off by the Smart Mirror's Raspberry Pi. Doing so, this reduces the power consumption by more than 50%. The smart mirror benefits the system as well because it allows the system to be working via remote desktop. This fixes the complications of having to open and close the box, reconnecting the HDMI cable, and having to log onto the computer nearest the Odroid. As of right now, the remote desktop feature has been addressed and understood to be able to implement onto the Raspberry Pi. The next steps for the team include installation of the magic mirror, configuring the remote desktop settings, and finding a way to receive the energy consumption data from AK's building engineer. The team looks forward to the challenge.

## 6. Appendix

### 6.1 Power Panels User Manual

Operation and Procedure Description for the Atwater Kent Power Panel:

1. Go to AK 113 and use a screwdriver to open the plastic box.
2. In the plastic box, there is a VGA hdmi converter. Using a long hdmi cable and the VGA hdmi converter connect the ODROID to the computer monitor. Disconnect the mouse and keyboard from the closest lab bench and connect it to the powered usb hub inside the box. This will be used for navigating through the ODROID.
3. Turn on the bottom switch (This switch powers everything that is within the small box, the ODROID, powered usb hub, and sending card.
4. On the computer monitor you should see an Ubuntu welcome screen. You should be automatically logged into odroid. You can change the setting by following the advanced user setting later in the appendix if you want to change the automatic login.
5. You will be see the screen display below. For the terminal, type in exit to close it. And simply choose to cancel the warning.



*Figure 41. Odroid Starting Display*

6. Ensure that the ODROID is connected to the internet, preferably through Ethernet. We used our student account to register the odroid as a device so repeat the same process if the odroid is not getting recognized after our account are closed.

7. (Optional) On the desktop there is a folder called Start app, which has 2 additional folders inside of it. Opening them will give shortcut to the power panel and children panel. They are the exported application of the processing sketches. Double click and choose run on the notification window to show the displays. This only displays the output.

8. To access the code, open processing from desktop. You will see a diamond icon with a program in it called processing. Right click and hit OPEN. Then click RUN on the notification window. Processing should begin to open. Once it is opened to File->Open->desktop->Children\_Panels 4-27b->Children\_Panels->Children\_Panels.pde

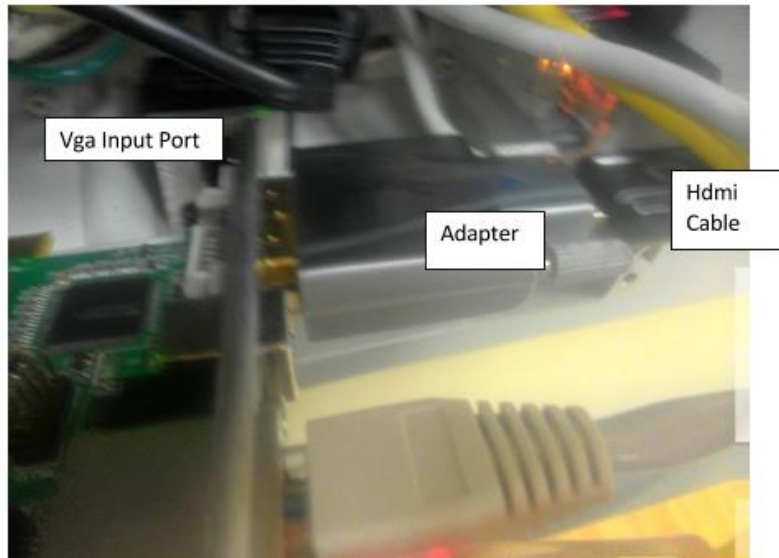
9. The code should open up, and you can then go to the top left of the window to the run button.

10. Next do the same with the Power Panel. Go to File->Open->Desktop->PowerPanel 3-27b -> PowerPanel -> PowerPanel.pde. The code should open. Do the same where you hit the run button like in step 9. This should open the Power Panel window.

11. The children panel and power panel windows should automatically position themselves in the correct locations. Before moving forward click on the PowerPanel window using the mouse, anywhere, just to make sure that it is the currently active window to respond to keyboard presses and inputs.

12. Once that is done you may disconnect the long hdmi and vga hdmi converter from the monitor and the ODROID. Take the small hdmi, attach it to the vga hdmi adapter and then connect it to the ODROID. Then take the end with the vga hdmi adapter and connect it to the sending card. As shown in the picture below.





*Figure 42. Sending card ports*

13. Once it is connected you may flip the TOP switch which turn on the power panels in the pumpkin lounge.

14. To turn off the system it would be best to turn off the ODROID properly through its menu, rather than just flipping the switch off. This can cause corruption issues with the sd card. It may be a good idea to look into making a copy of the image for back up.

## 6.2 Making System autonomous

- One of the project goals was to build this system to be easily operated by WPI Crimson Key. The Crimson Key Tour Guides are a group of students who volunteer their time to giving tours and sharing their stories to prospective students and families.
- In order to make the system autonomous, the team attempted to find ways to load up the processing displays from simply turning the system on. During this attempt, the processing sketches were exported. Below is a figure that shows the Export option in processing which was saved in the ODROID's desktop as application shortcuts.

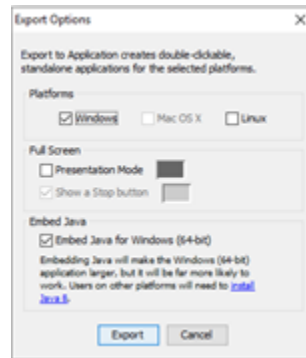


Figure 43. Export Option in 'Processing'



Figure 44. Exported Folders for Children and Power Panel Sketches in Desktop

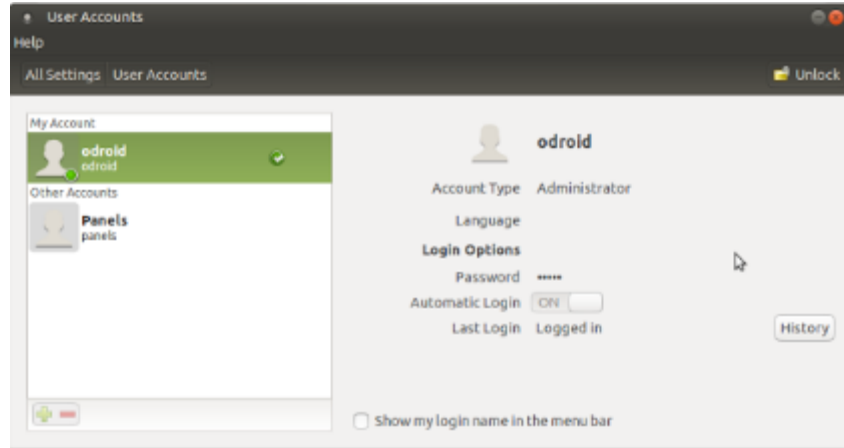


Figure 45. Application shortcut for Children Panels Display



*Figure 46. Application shortcut for Power Panels Display*

- This provided an application shortcut which would load up the panel displays directly. Next, the team tried to include that in the Ubuntu built in startup program.
- Including the shortcut in the built in startup did not yield any result as the sketch display did not start when the system turned on. The team decided to look at other options.
- It was discovered that a “systemd” which is built-in for Ubuntu 15.04. “Systemd” is a system and service manager for Linux operating systems and starts the rest of the system. It can be used to run scripts which could take in the source files of the sketch shortcuts and run it right from boot. Unfortunately, the team did not have any prior knowledge in understanding the usage of “systemd” and scripts. It was then attempted to follow online research to work with “systemd” but was unsuccessful.
- The team also looked at the unity-control center to automatically log in another new user called ‘Panels’ with less privileges. There is more information in the appendix under accessing advanced user settings. As it is an automatic login, the team did not want just any user to be able to make admin changes to the ODROID which is why panels was created.



*Figure 47. Automatic login option in advanced user setting*

- The team attempted to move only the processing application and the panels code to the 'Panels' from the shared public folder within the ODROID but there was no access to it. The next option was to use a flash drive to move the files but it only worked with user 'odroid'. 'Panel' was unable to recognize the flash drive as a file system.
- The control for the autonomous panels should be the smart mirror. If the processing sketches can be displayed after booting up, a switch should be used with the smart mirror in order to turn on the system. A physical switch can be extended to the smart mirror through the wall but according to Professor O'rourke, there is a lot of steps and time required to get approval from the electricians in WPI to add such a switch. It also would not be a good option until the power issue of the system is taken care of.





Figure 50. Processing sketch output in desktop

## 6.4 Accessing Advanced User settings

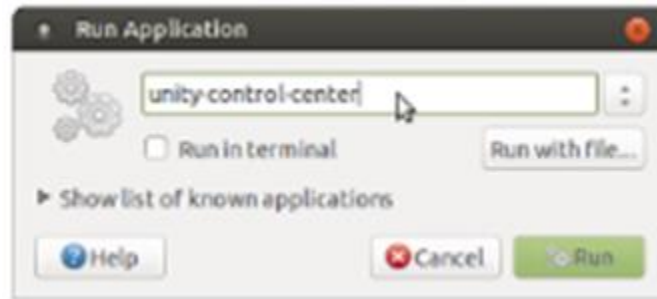


Figure 51. Shortcut to unity control center

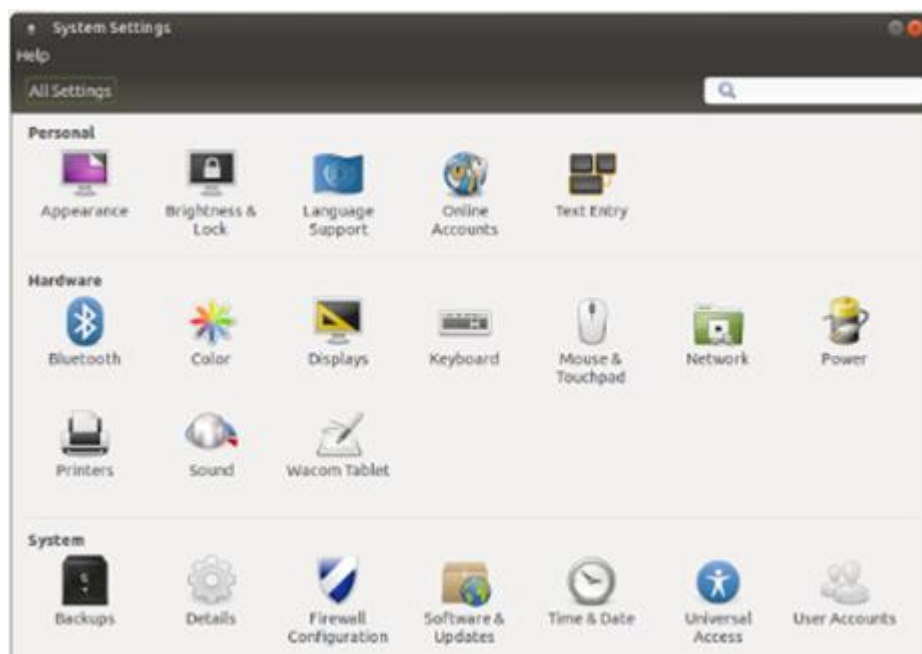


Figure 52. Gnome System Settings

- Instruction:
  - Press Alt + F2 to access user setting in gnome center.
  - Type in: unity-control-center
- This displays the old menu system for the Ubuntu system. Some of the settings do not work as the controls are no longer functional such as “Display”. But the user settings option still function. It allows for one to automatically login a user at startup which is a functionality that was needed for making the system autonomous.

## 6.5 LED Studio Software Manual

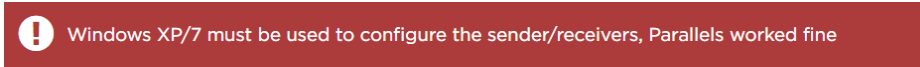
Operation and Procedure Description for the LED Studio Software:

1. Download the software .exe file from the following link:

<https://learn.adafruit.com/adafruit-diy-led-video-wall/led-studio-software>

LED Studio Software Configuration

by James DeVito



In order to configure the LED panels you will need to download the Linsn LED studio software. It's free, however they will ask for a serial number- just enter "888888".

LEDStudio12.23.exe

We also suggest our LED wall configuration files

Download the Adawall Config

If you are using 32x32 or 32x64 panels, check out the config file from the LED cube here!

*Figure 53. Adafruit LED Studio Software Download*

If the software does not work after completing the procedures, use this link instead and download the latest version of the software:

<https://www.vegasledscreens.com/downloads/category/1-led-studio-software.html>

A screenshot of a software download page for "LED Studio Software V12.60". The page has a dark background with white text. It includes a title "LED Studio Software", a description of the software, file name "Ledstudio-1260-1221.exe", file size "54.65 MB", version "v12.60", author "Vegas LED Screens", email "info[at]vegasledscreens[dot]com", and date "23. May 2016". There is a "Description" section with more details. At the bottom, there is a "Download" button and a "Like 53" button. A "G+1" button is also visible.

*Figure 54. Recent Version of LED Studio Software*

2. Using the first link from step one, proceed to using the following USB cable to connect the LED video wall controller to the device the LED Studio Software is installed on.





*Figure 55. USB cable to LED video wall controller*

3. Run the software and it will prompt the user with the following message

A screenshot of a Windows-style dialog box titled "LED software11". The dialog box has a dark blue header bar with the title and a close button. Below the header, the text "Registration Information" is displayed. A small CD-ROM icon is visible in the top right corner. The main area of the dialog box contains the following text: "Please enter the name and company of the registered owner of LED software11 into the fields below. All fields must be filled in to proceed." Below this text are three input fields: "Name:" with the value "James", "Company:" with the value "Adafruit", and "Serial:" with the value "888888". At the bottom of the dialog box, there is a progress bar labeled "LED software11 Installation" and three buttons: "< Back", "Next >", and "Cancel".

*Figure 56. Registration Information*

The name and company can be entered with anything, but the serial number is always “888888”

4. Click “option” then “software setup”

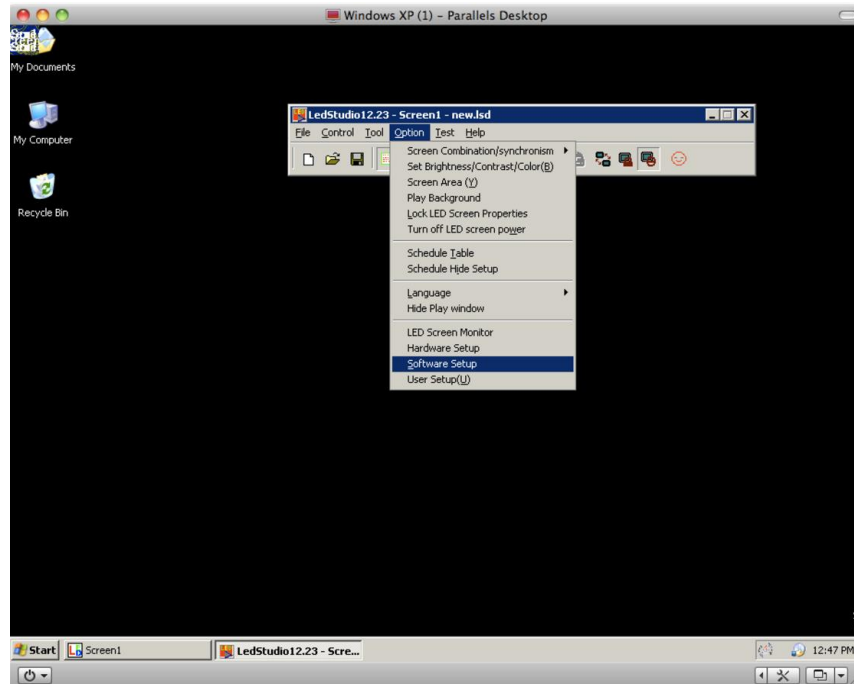


Figure 57. Software Setup

5. Once in the “software setup” of the program, type “linsn” anywhere and a text will ask for a password. Type “168” to proceed.

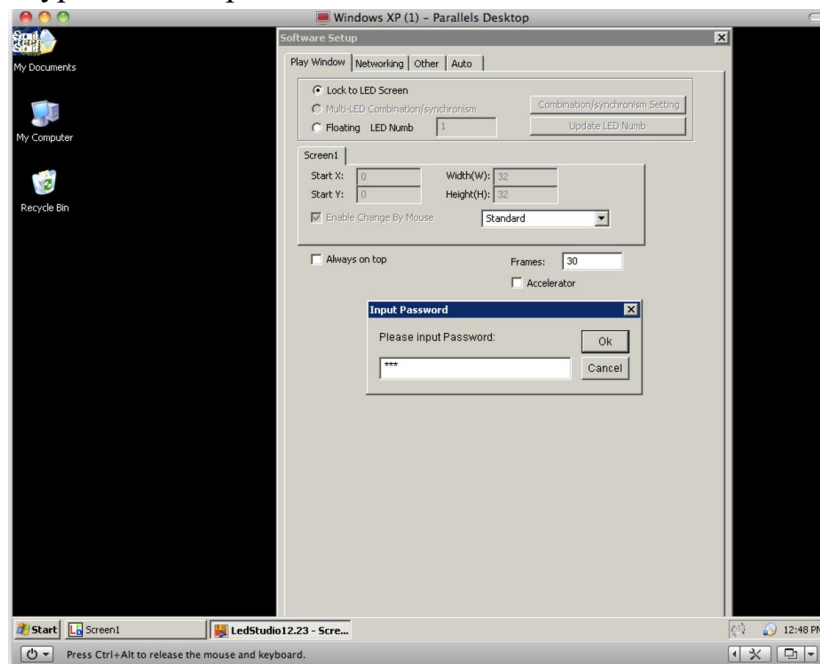


Figure 58. Software Setup Input Password

6. This will present the sender tab where users can adjust the display resolution, mirror/rotation, as well the start X/Y position of what part of the screen will display on the LED wall. This will update in realtime. Click Save on Sender when changes to the settings are complete.

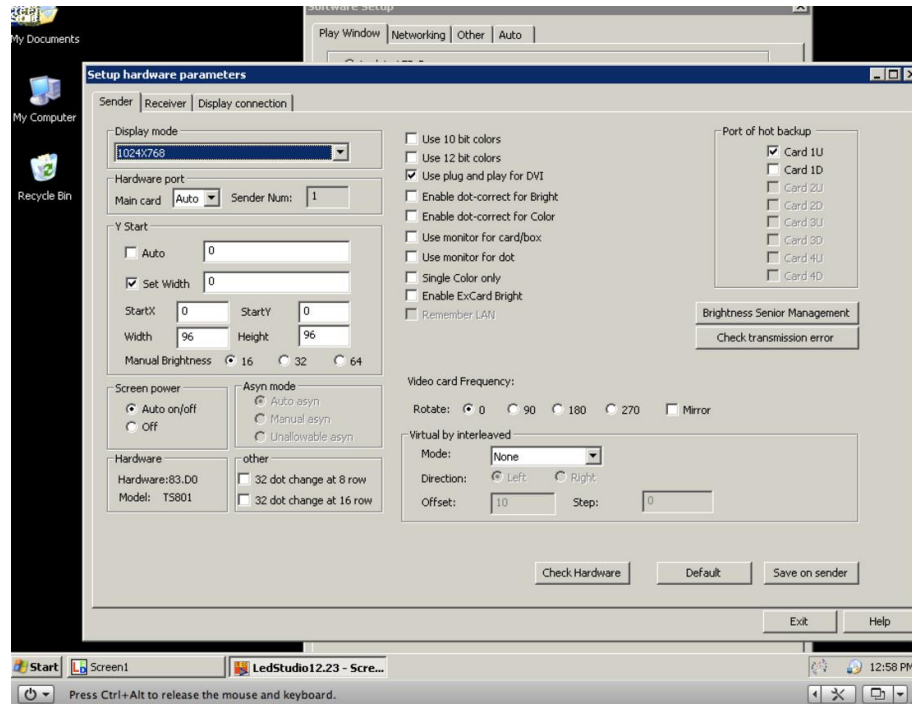


Figure 59. Sender Card Settings

7. If users wish to make complex changes to the receiver tab, then download the configuration file from the first step and proceed to loading it onto the tab. This opens up more options to configure multiple receiver cards and their displays, but does require users to have advanced knowledge in this area to work with these options. They are not necessary to run the LED Panels as they are, but can provide more choices to alter the displays of the LED Panels.

## 6.6 System Power Issues

- Background –

The system has LED panels which are matrixes of LED panels combined together. The configuration software is used to control the display of the panels and match it with a group of panel matrixes. The table below shows the power demand for the system with just the LED panels displaying white in full brightness. Analyzing the table reveals that the LED panels draw up too much current when they are turned on.

Display	Number of Panels	Total Power Draw [W]	Total Current Draw [A]
Main	36	720	144
Horizontal Ticker	12	240	48
Vertical Ticker	4	80	16
Large Child	4	80	16
Small Child	2	40	8
Total	64	1280	256

*Table 3. Summary of Power and Current Demand for each panel display*

- Limitations –

From Professor O'Rourke, it was noted that AK 113 where the panels get their power was not originally meant to be used as a computer lab room. This in turn meant that the room did not have a circuit breaker designed to handle the computers and the LED panel system. The panels and 3 of the computers within the lab room are connected to one circuit breaker. As the panels are a high current system, they are not the most stable system. There have been a few times where the system has tripped the circuit breaker. As AK 113 is busy with students working on all kinds of labs, accidentally shorting the system and activating the circuit breaker can cause some of them to lose their work progress. The whole system is meant to be helping students and visitors so it is best not to compromise that. Additionally, the team does not want to have the circuit breaker activate since it is an indication of a faulty system design which could overload the system and damage a lifetime of systems connected in that circuit breaker.

- Smart mirror issue –

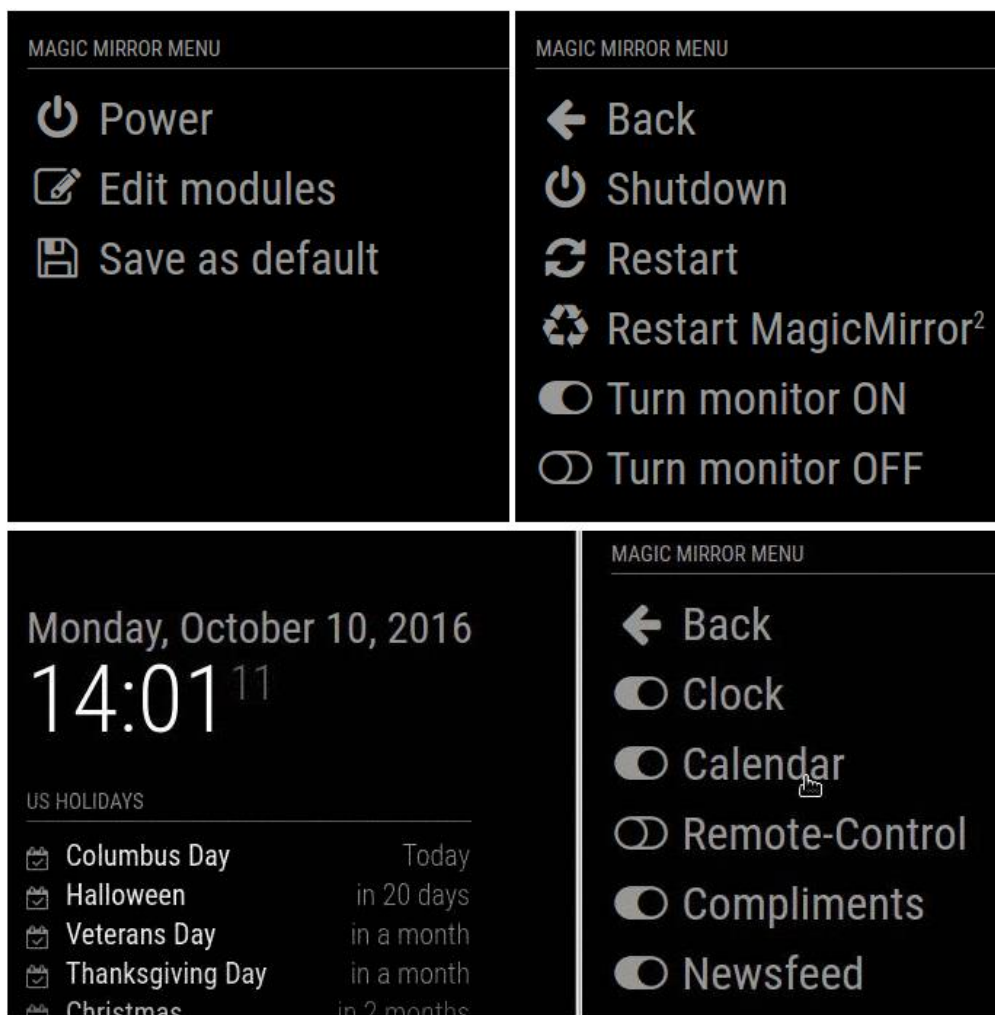
The Smart Mirror is included in the same circuit breaker because of its location. It was noticed that with the panels turned on, the Smart Mirror cannot power on. The current draw from our system is probably too strong to properly supply the smart mirror. The solution for is use of an extension cord to connect it to a different circuit to avoid adding more load in the LED panel circuit.

- Takeaway

Our LED panel system uses a lot of power and is connected to a circuit with other loads (computer benches in the lab) that wasn't designed to handle such power level. It is thus not recommended to have the system on for extended period of time as it could trip the circuit breaker and cut power for the lab bench where students could be working. Testing the system should be done over the weekend when there are less students with permission from the school. It can be used to better understanding of the system, its limits and find solution to the problems.

## 6.7 Smart Mirror User Manual

The easiest way to turning the Smart Mirror on or off is to turn on/off the power supply. However, by turning the power off “unexpectedly” the SD Card of the Raspberry Pi can become corrupted. The alternative method that our team has includes the use of a website. This module was uploaded from the Magic Mirror Forum called: MMM-Remote Control. It allows authorized users to be able to have an easier interface for editing properties of the Smart Mirror itself.



*Figure 60. Magic Mirror Module: Remote Control*

This picture below are highlights of the user website. From this website, one will be able to control the power, module, brightness, and editing settings. The website includes the

IP address of the Raspberry Pi and hence will not be included in this user manual. The official operation manual given to ECE Technician, Bill Appleyard or to Professor Mazumder.

The following are instructions on how to download and run the module based on the work of user: Jopyth on the Magic Mirror Forum.

1) Clone this repository in your modules folder, and install dependencies:

```
cd ~/MagicMirror/modules # adapt directory if you are
using a different one
git clone https://github.com/Jopyth/MMM-Remote-
Control.git
cd MMM-Remote-Control
npm install
```

2) Add the module to your config/config.js file, if you add a position, it will display the URL to the remote on the mirror.

```
{
  module: 'MMM-Remote-Control'
  // uncomment the following line to show the URL of
  the remote control on the mirror
  // , position: 'bottom_left'
  // you can hide this module afterwards from the
  remote control itself
},
```

3) Add the IP addresses of devices you want to use to access the Remote Control to the ipWhiteList in your config.js.

4) Restart your Magic Mirror<sup>2</sup> (i.e. pm2 restart mm).

5) Access the remote interface on <http://192.168.xxx.xxx:8080/remote.html> (replace with IP address of your RaspberryPi).

Note: If your user does not have sudo rights, the shutdown does not work (it *should* work for everyone who did not change anything on this matter).

## 6.8 Code from Smart Mirror

Magic Mirror is an online readily available program designed by students at MIT. Through this website, hundreds of users have been able to design and modify their own module. A module is a component feature for the Smart Mirror to display. There is a section in the MQP report explaining how to install and use Magic Mirror. Below is the default code for the default Modules (the modules that come with you initially install Magic Mirror):

### Default Modules:

#### Clock

```
// Module config defaults.
defaults: {
  displayType: "digital", // options: digital, analog, both

  timeFormat: config.timeFormat,
  displaySeconds: true,
  showPeriod: true,
  showPeriodUpper: false,
  clockBold: false,
  showDate: true,
  showWeek: false,
  dateFormat: "dddd, LL",

  /* specific to the analog clock */
  analogSize: "200px",
  analogFace: "simple", // options: 'none', 'simple', 'face-###' (where ### is 001
  to 012 inclusive)
  analogPlacement: "bottom", // options: 'top', 'bottom', 'left', 'right'
  analogShowDate: "top", // options: false, 'top', or 'bottom'
  secondsColor: "#888888",
  timezone: null,
},
// Define required scripts.
getScripts: function() {
  return ["moment.js", "moment-timezone.js"];
},
// Define styles.
getStyles: function() {
```



```

return ["clock_styles.css"];
},
// Define start sequence.
start: function() {
Log.info("Starting module: " + this.name);

// Schedule update interval.
var self = this;
setInterval(function() {
self.updateDom();
}, 1000);

// Set locale.
moment.locale(config.language);

},
// Override dom generator.
getDom: function() {

var wrapper = document.createElement("div");

/*****
* Create wrappers for DIGITAL clock
*/

var dateWrapper = document.createElement("div");
var timeWrapper = document.createElement("div");
var secondsWrapper = document.createElement("sup");
var periodWrapper = document.createElement("span");
var weekWrapper = document.createElement("div")
// Style Wrappers
dateWrapper.className = "date normal medium";
timeWrapper.className = "time bright large light";
secondsWrapper.className = "dimmed";
weekWrapper.className = "week dimmed medium"

// Set content of wrappers.
// The moment().format("h") method has a bug on the Raspberry Pi.
// So we need to generate the timestring manually.
// See issue: https://github.com/MichMich/MagicMirror/issues/181
var timeString;
var now = moment();
if (this.config.timezone) {
now.tz(this.config.timezone);
}
}

```

```

var hourSymbol = "HH";
if (this.config.timeFormat !== 24) {
hourSymbol = "h";
}

if (this.config.clockBold === true) {
timeString = now.format(hourSymbol + "[<span class=\"bold\">]mm[</span>]");
} else {
timeString = now.format(hourSymbol + ":mm");
}

if(this.config.showDate){
dateWrapper.innerHTML = now.format(this.config.dateFormat);
}
if (this.config.showWeek) {
weekWrapper.innerHTML = this.translate("WEEK") + " " + now.week();
}
timeWrapper.innerHTML = timeString;
secondsWrapper.innerHTML = now.format("ss");
if (this.config.showPeriodUpper) {
periodWrapper.innerHTML = now.format("A");
} else {
periodWrapper.innerHTML = now.format("a");
}
if (this.config.displaySeconds) {
timeWrapper.appendChild(secondsWrapper);
}
if (this.config.showPeriod && this.config.timeFormat !== 24) {
timeWrapper.appendChild(periodWrapper);
}

/*****
* Create wrappers for ANALOG clock, only if specified in config
*/

if (this.config.displayType !== "digital") {
// If it isn't 'digital', then an 'analog' clock was also requested

// Calculate the degree offset for each hand of the clock
var now = moment();
if (this.config.timezone) {
now.tz(this.config.timezone);
}
var second = now.seconds() * 6,

```

```
minute = now.minute() * 6 + second / 60,
hour = ((now.hours() % 12) / 12) * 360 + 90 + minute / 12;

// Create wrappers
var wrapper = document.createElement("div");
var clockCircle = document.createElement("div");
clockCircle.className = "clockCircle";
clockCircle.style.width = this.config.analogSize;
clockCircle.style.height = this.config.analogSize;

if (this.config.analogFace != "" && this.config.analogFace != "simple" &&
this.config.analogFace != "none") {
clockCircle.style.background = "url("+ this.data.path + "faces/" +
this.config.analogFace + ".svg)";
clockCircle.style.backgroundSize = "100%";

// The following line solves issue:
https://github.com/MichMich/MagicMirror/issues/611
clockCircle.style.border = "1px solid black";

} else if (this.config.analogFace != "none") {
clockCircle.style.border = "2px solid white";
}

var clockFace = document.createElement("div");
clockFace.className = "clockFace";

var clockHour = document.createElement("div");
clockHour.id = "clockHour";
clockHour.style.transform = "rotate(" + hour + "deg)";
clockHour.className = "clockHour";
var clockMinute = document.createElement("div");
clockMinute.id = "clockMinute";
clockMinute.style.transform = "rotate(" + minute + "deg)";
clockMinute.className = "clockMinute";

// Combine analog wrappers
clockFace.appendChild(clockHour);
clockFace.appendChild(clockMinute);

if (this.config.displaySeconds) {
var clockSecond = document.createElement("div");
clockSecond.id = "clockSecond";
clockSecond.style.transform = "rotate(" + second + "deg)";
clockSecond.className = "clockSecond";
clockSecond.style.backgroundColor = this.config.secondsColor;
clockFace.appendChild(clockSecond);
}
```

```

}
clockCircle.appendChild(clockFace);
}

/*****
* Combine wrappers, check for .displayType
*/

if (this.config.displayType === "digital") {
// Display only a digital clock
wrapper.appendChild(dateWrapper);
wrapper.appendChild(timeWrapper);
wrapper.appendChild(weekWrapper);
} else if (this.config.displayType === "analog") {
// Display only an analog clock
dateWrapper.style.textAlign = "center";

if (this.config.showWeek) {
weekWrapper.style.paddingBottom = "15px";
} else {
dateWrapper.style.paddingBottom = "15px";
}

if (this.config.analogShowDate === "top") {
wrapper.appendChild(dateWrapper);
wrapper.appendChild(weekWrapper);
wrapper.appendChild(clockCircle);
} else if (this.config.analogShowDate === "bottom") {
wrapper.appendChild(clockCircle);
wrapper.appendChild(dateWrapper);
wrapper.appendChild(weekWrapper);
} else {
wrapper.appendChild(clockCircle);
}
} else {
// Both clocks have been configured, check position
var placement = this.config.analogPlacement;

analogWrapper = document.createElement("div");
analogWrapper.id = "analog";
analogWrapper.style.cssFloat = "none";
analogWrapper.appendChild(clockCircle);
digitalWrapper = document.createElement("div");
digitalWrapper.id = "digital";
digitalWrapper.style.cssFloat = "none";

```

```

digitalWrapper.appendChild(dateWrapper);
digitalWrapper.appendChild(timeWrapper);
digitalWrapper.appendChild(weekWrapper);

var appendClocks = function(condition, pos1, pos2) {
var padding = [0,0,0,0];
padding[(placement === condition) ? pos1 : pos2] = "20px";
analogWrapper.style.padding = padding.join(" ");
if (placement === condition) {
wrapper.appendChild(analogWrapper);
wrapper.appendChild(digitalWrapper);
} else {
wrapper.appendChild(digitalWrapper);
wrapper.appendChild(analogWrapper);
}
};

if (placement === "left" || placement === "right") {
digitalWrapper.style.display = "inline-block";
digitalWrapper.style.verticalAlign = "top";
analogWrapper.style.display = "inline-block";

appendClocks("left", 1, 3);
} else {
digitalWrapper.style.textAlign = "center";

appendClocks("top", 2, 0);
}

// Return the wrapper to the dom.
return wrapper;
}
});

```

**To use this module, add it to the modules array in the `config/config.js` file:**

```

modules: [
  {
    module: "clock",
    position: "top_left",      // This can be any of the regions.
    config: {
      // The config property is optional.
      // See 'Configuration options' for more information.
    }
  }
]

```

]

To edit this file:

<https://github.com/MichMich/MagicMirror/tree/master/modules/default/clock>

## Calendar

```
Module.register("calender", {

  // Define module defaults
  defaults: {
    maximumEntries: 10, // Total Maximum Entries
    maximumNumberOfDays: 365,
    displaySymbol: true,
    defaultSymbol: "calendar", // Fontawesome Symbol see
    http://fontawesome.io/cheatsheet/
    displayRepeatingCountTitle: false,
    defaultRepeatingCountTitle: "",
    maxTitleLength: 25,
    fetchInterval: 5 * 60 * 1000, // Update every 5 minutes.
    animationSpeed: 2000,
    fade: true,
    urgency: 7,
    timeFormat: "relative",
    dateFormat: "MMM Do",
    getRelative: 6,
    fadePoint: 0.25, // Start on 1/4th of the list.
    hidePrivate: false,
    colored: false,
    calendars: [
      {
        symbol: "calendar",
        url: "http://www.calendarlabs.com/templates/ical/US-Holidays.ics",
      },
    ],
    titleReplace: {
      "De verjaardag van ": "",
      "'s birthday": ""
    },
    broadcastEvents: true,
    excludedEvents: []
  },

  // Define required scripts.
  getStyles: function () {
    return ["calendar.css", "font-awesome.css"];
  },

  // Define required scripts.
  getScripts: function () {
    return ["moment.js"];
  },
```

```

// Define required translations.
getTranslations: function () {
// The translations for the default modules are defined in the core translation
files.
// Therefor we can just return false. Otherwise we should have returned a
dictionary.
// If you're trying to build your own module including translations, check out
the documentation.
return false;
},

// Override start method.
start: function () {
Log.log("Starting module: " + this.name);

// Set locale.
moment.locale(config.language);

for (var c in this.config.calendars) {
var calendar = this.config.calendars[c];
calendar.url = calendar.url.replace("webcal://", "http://");

var calendarConfig = {
maximumEntries: calendar.maximumEntries,
maximumNumberOfDays: calendar.maximumNumberOfDays
};

// we check user and password here for backwards compatibility with old configs
if(calendar.user && calendar.pass){
calendar.auth = {
user: calendar.user,
pass: calendar.pass
}
}

this.addCalendar(calendar.url, calendar.auth, calendarConfig);
}

this.calendarData = {};
this.loaded = false;
},

// Override socket notification handler.
socketNotificationReceived: function (notification, payload) {
if (notification === "CALENDAR_EVENTS") {
if (this.hasCalendarURL(payload.url)) {
this.calendarData[payload.url] = payload.events;
this.loaded = true;

if (this.config.broadcastEvents) {
this.broadcastEvents();
}
}
} else if (notification === "FETCH_ERROR") {
Log.error("Calendar Error. Could not fetch calendar: " + payload.url);
}
}
}

```

```

} else if (notification === "INCORRECT_URL") {
Log.error("Calendar Error. Incorrect url: " + payload.url);
} else {
Log.log("Calendar received an unknown socket notification: " + notification);
}

this.updateDom(this.config.animationSpeed);
},

// Override dom generator.
getDom: function () {

var events = this.createEventList();
var wrapper = document.createElement("table");
wrapper.className = "small";

if (events.length === 0) {
wrapper.innerHTML = (this.loaded) ? this.translate("EMPTY") :
this.translate("LOADING");
wrapper.className = "small dimmed";
return wrapper;
}

for (var e in events) {
var event = events[e];

var excluded = false;
for (var f in this.config.excludedEvents) {
var filter = this.config.excludedEvents[f];
if (event.title.toLowerCase().includes(filter.toLowerCase())) {
excluded = true;
break;
}
}

if (excluded) {
continue;
}

var eventWrapper = document.createElement("tr");

if (this.config.colored) {
eventWrapper.style.cssText = "color:" + this.colorForUrl(event.url);
}

eventWrapper.className = "normal";

if (this.config.displaySymbol) {
var symbolWrapper = document.createElement("td");
symbolWrapper.className = "symbol align-right";
var symbols = this.symbolsForUrl(event.url);
if(typeof symbols === "string") {
symbols = [symbols];
}
}

```



```

for(var i = 0; i < symbols.length; i++) {
var symbol = document.createElement("span");
symbol.className = "fa fa-" + symbols[i];
if(i > 0){
symbol.style.paddingLeft = "5px";
}
symbolWrapper.appendChild(symbol);
}
eventWrapper.appendChild(symbolWrapper);
}

var titleWrapper = document.createElement("td"),
repeatingCountTitle = "";

if (this.config.displayRepeatingCountTitle) {

repeatingCountTitle = this.countTitleForUrl(event.url);

if (repeatingCountTitle !== "") {
var thisYear = new Date(parseInt(event.startDate)).getFullYear(),
yearDiff = thisYear - event.firstYear;

repeatingCountTitle = ", " + yearDiff + ". " + repeatingCountTitle;
}
}

titleWrapper.innerHTML = this.titleTransform(event.title) + repeatingCountTitle;

if (!this.config.colored) {
titleWrapper.className = "title bright";
} else {
titleWrapper.className = "title";
}

eventWrapper.appendChild(titleWrapper);

var timeWrapper = document.createElement("td");
//console.log(event.today);
var now = new Date();
// Define second, minute, hour, and day variables
var oneSecond = 1000; // 1,000 milliseconds
var oneMinute = oneSecond * 60;
var oneHour = oneMinute * 60;
var oneDay = oneHour * 24;
if (event.fullDayEvent) {
if (event.today) {
timeWrapper.innerHTML = this.capFirst(this.translate("TODAY"));
} else if (event.startDate - now < oneDay && event.startDate - now > 0) {
timeWrapper.innerHTML = this.capFirst(this.translate("TOMORROW"));
} else if (event.startDate - now < 2 * oneDay && event.startDate - now > 0) {
if (this.translate("DAYAFTERTOMORROW") !== "DAYAFTERTOMORROW") {
timeWrapper.innerHTML = this.capFirst(this.translate("DAYAFTERTOMORROW"));
} else {
timeWrapper.innerHTML = this.capFirst(moment(event.startDate, "x").fromNow());
}
}
}
}

```

```

}
} else {
/* Check to see if the user displays absolute or relative dates with their events
* Also check to see if an event is happening within an 'urgency' time
frameElement
* For example, if the user set an .urgency of 7 days, those events that fall
within that
* time frame will be displayed with 'in xxx' time format or moment.fromNow()
*
* Note: this needs to be put in its own function, as the whole thing repeats
again verbatim
*/
if (this.config.timeFormat === "absolute") {
if ((this.config.urgency > 1) && (event.startDate - now < (this.config.urgency *
oneDay))) {
// This event falls within the config.urgency period that the user has set
timeWrapper.innerHTML = this.capFirst(moment(event.startDate, "x").fromNow());
} else {
timeWrapper.innerHTML = this.capFirst(moment(event.startDate,
"x").format(this.config.dateFormat));
}
} else {
timeWrapper.innerHTML = this.capFirst(moment(event.startDate, "x").fromNow());
}
}
} else {

if (event.startDate >= new Date()) {
if (event.startDate - now < 2 * oneDay) {
// This event is within the next 48 hours (2 days)
if (event.startDate - now < this.config.getRelative * oneHour) {
// If event is within 6 hour, display 'in xxx' time format or moment.fromNow()
timeWrapper.innerHTML = this.capFirst(moment(event.startDate, "x").fromNow());
} else {
// Otherwise just say 'Today/Tomorrow at such-n-such time'
timeWrapper.innerHTML = this.capFirst(moment(event.startDate, "x").calendar());
}
} else {
/* Check to see if the user displays absolute or relative dates with their events
* Also check to see if an event is happening within an 'urgency' time
frameElement
* For example, if the user set an .urgency of 7 days, those events that fall
within that
* time frame will be displayed with 'in xxx' time format or moment.fromNow()
*
* Note: this needs to be put in its own function, as the whole thing repeats
again verbatim
*/
if (this.config.timeFormat === "absolute") {
if ((this.config.urgency > 1) && (event.startDate - now < (this.config.urgency *
oneDay))) {
// This event falls within the config.urgency period that the user has set
timeWrapper.innerHTML = this.capFirst(moment(event.startDate, "x").fromNow());
} else {

```

```

timeWrapper.innerHTML = this.capFirst(moment(event.startDate,
"x").format(this.config.dateFormat));
}
} else {
timeWrapper.innerHTML = this.capFirst(moment(event.startDate, "x").fromNow());
}
}
} else {
timeWrapper.innerHTML = this.capFirst(this.translate("RUNNING")) + " " +
moment(event.endDate, "x").fromNow(true);
}
}
//timeWrapper.innerHTML += ' - ' + moment(event.startDate, 'x').format('lll');
//console.log(event);
timeWrapper.className = "time light";
eventWrapper.appendChild(timeWrapper);

wrapper.appendChild(eventWrapper);

// Create fade effect.
if (this.config.fade && this.config.fadePoint < 1) {
if (this.config.fadePoint < 0) {
this.config.fadePoint = 0;
}
var startingPoint = events.length * this.config.fadePoint;
var steps = events.length - startingPoint;
if (e >= startingPoint) {
var currentStep = e - startingPoint;
eventWrapper.style.opacity = 1 - (1 / steps * currentStep);
}
}
}

return wrapper;
},

/* hasCalendarURL(url)
* Check if this config contains the calendar url.
*
* argument url string - Url to look for.
*
* return bool - Has calendar url
*/
hasCalendarURL: function (url) {
for (var c in this.config.calendars) {
var calendar = this.config.calendars[c];
if (calendar.url === url) {
return true;
}
}

return false;
},

/* createEventList()

```

```

* Creates the sorted list of all events.
*
* return array - Array with events.
*/
createEventList: function () {
var events = [];
var today = moment().startOf("day");
for (var c in this.calendarData) {
var calendar = this.calendarData[c];
for (var e in calendar) {
var event = calendar[e];
if(this.config.hidePrivate) {
if(event.class === "PRIVATE") {
// do not add the current event, skip it
continue;
}
}
event.url = c;
event.today = event.startDate >= today && event.startDate < (today + 24 * 60 * 60
* 1000);
events.push(event);
}
}

events.sort(function (a, b) {
return a.startDate - b.startDate;
});

return events.slice(0, this.config.maximumEntries);
},

/* createEventList(url)
* Requests node helper to add calendar url.
*
* argument url string - Url to add.
*/
addCalendar: function (url, auth, calendarConfig) {
this.sendSocketNotification("ADD_CALENDAR", {
url: url,
maximumEntries: calendarConfig.maximumEntries || this.config.maximumEntries,
maximumNumberOfDays: calendarConfig.maximumNumberOfDays ||
this.config.maximumNumberOfDays,
fetchInterval: this.config.fetchInterval,
auth: auth
});
},

/* symbolsForUrl(url)
* Retrieves the symbols for a specific url.
*
* argument url string - Url to look for.
*
* return string/array - The Symbols
*/
symbolsForUrl: function (url) {

```

```

return this.getCalendarProperty(url, "symbol", this.config.defaultSymbol);
},

/* colorForUrl(url)
 * Retrieves the color for a specific url.
 *
 * argument url string - Url to look for.
 *
 * return string - The Color
 */
colorForUrl: function (url) {
return this.getCalendarProperty(url, "color", "#fff");
},

/* countTitleForUrl(url)
 * Retrieves the name for a specific url.
 *
 * argument url string - Url to look for.
 *
 * return string - The Symbol
 */
countTitleForUrl: function (url) {
return this.getCalendarProperty(url, "repeatingCountTitle",
this.config.defaultRepeatingCountTitle);
},

/* getCalendarProperty(url, property, defaultValue)
 * Helper method to retrieve the property for a specific url.
 *
 * argument url string - Url to look for.
 * argument property string - Property to look for.
 * argument defaultValue string - Value if property is not found.
 *
 * return string - The Property
 */
getCalendarProperty: function (url, property, defaultValue) {
for (var c in this.config.calendars) {
var calendar = this.config.calendars[c];
if (calendar.url === url && calendar.hasOwnProperty(property)) {
return calendar[property];
}
}

return defaultValue;
},

/* shorten(string, maxLength)
 * Shortens a string if it's longer than maxLength.
 * Adds an ellipsis to the end.
 *
 * argument string string - The string to shorten.
 * argument maxLength number - The max length of the string.
 *
 * return string - The shortened string.
 */

```

```

shorten: function (string, maxLength) {
  if (string.length > maxLength) {
    return string.slice(0, maxLength) + "&hellip;";
  }

  return string;
},

/* capFirst(string)
 * Capitalize the first letter of a string
 * Return capitalized string
 */

capFirst: function (string) {
  return string.charAt(0).toUpperCase() + string.slice(1);
},

/* titleTransform(title)
 * Transforms the title of an event for usage.
 * Replaces parts of the text as defined in config.titleReplace.
 * Shortens title based on config.maxTitleLength
 *
 * argument title string - The title to transform.
 *
 * return string - The transformed title.
 */
titleTransform: function (title) {
  for (var needle in this.config.titleReplace) {
    var replacement = this.config.titleReplace[needle];

    var regParts = needle.match(/^\/(.+)\\/([gim]*)$/);
    if (regParts) {
      // the parsed pattern is a regexp.
      needle = new RegExp(regParts[1], regParts[2]);
    }

    title = title.replace(needle, replacement);
  }

  title = this.shorten(title, this.config.maxTitleLength);
  return title;
},

/* broadcastEvents()
 * Broadcasts the events to all other modules for reuse.
 * The all events available in one array, sorted on startdate.
 */
broadcastEvents: function () {
  var eventList = [];
  for (url in this.calendarData) {
    var calendar = this.calendarData[url];
    for (e in calendar) {
      var event = cloneObject(calendar[e]);
      delete event.url;
      eventList.push(event);
    }
  }
}

```

```

    }
  }

  eventList.sort(function(a,b) {
    return a.startDate - b.startDate;
  });

  this.sendNotification("CALENDAR_EVENTS", eventList);

}
});

```

To use this module, add it to the modules array in the `config/config.js` file:

```

modules: [
  {
    module: "calendar",
    position: "top_left", // This can be any of the regions. Best
    results in left or right regions.
    config: {
      // The config property is optional.
      // If no config is set, an example calendar is shown.
      // See 'Configuration options' for more information.
    }
  }
]

```

To edit this file:

<https://github.com/MichMich/MagicMirror/tree/master/modules/default/calendar>

## Current Weather

```

Module.register("currentweather",{

  // Default module config.
  defaults: {
    location: false,
    locationID: false,
    appid: "",
    units: config.units,
    updateInterval: 10 * 60 * 1000, // every 10 minutes
    animationSpeed: 1000,
    timeFormat: config.timeFormat,
    showPeriod: true,
    showPeriodUpper: false,
    showWindDirection: true,
    useBeaufort: true,
    lang: config.language,

```

```

showHumidity: false,
degreeLabel: false,

initialLoadDelay: 0, // 0 seconds delay
retryDelay: 2500,

apiVersion: "2.5",
apiBase: "http://api.openweathermap.org/data/",
weatherEndpoint: "weather",

appendLocationNameToHeader: true,
calendarClass: "calendar",

onlyTemp: false,
roundTemp: false,

iconTable: {
  "01d": "wi-day-sunny",
  "02d": "wi-day-cloudy",
  "03d": "wi-cloudy",
  "04d": "wi-cloudy-windy",
  "09d": "wi-showers",
  "10d": "wi-rain",
  "11d": "wi-thunderstorm",
  "13d": "wi-snow",
  "50d": "wi-fog",
  "01n": "wi-night-clear",
  "02n": "wi-night-cloudy",
  "03n": "wi-night-cloudy",
  "04n": "wi-night-cloudy",
  "09n": "wi-night-showers",
  "10n": "wi-night-rain",
  "11n": "wi-night-thunderstorm",
  "13n": "wi-night-snow",
  "50n": "wi-night-alt-cloudy-windy"
},
},

// create a variable for the first upcoming calendar event. Used if no location
is specified.
firstEvent: false,

// create a variable to hold the location name based on the API result.
fetchedLocationName: "",

// Define required scripts.
getScripts: function() {
  return ["moment.js"];
},

// Define required styles.
getStyles: function() {
  return ["weather-icons.css", "currentweather.css"];
},

```



```

// Define required translations.
getTranslations: function() {
// The translations for the default modules are defined in the core translation
files.
// Therefor we can just return false. Otherwise we should have returned a
dictionary.
// If you're trying to build your own module including translations, check out
the documentation.
return false;
},

// Define start sequence.
start: function() {
Log.info("Starting module: " + this.name);

// Set locale.
moment.locale(config.language);

this.windSpeed = null;
this.windDirection = null;
this.sunriseSunsetTime = null;
this.sunriseSunsetIcon = null;
this.temperature = null;
this.weatherType = null;

this.loaded = false;
this.scheduleUpdate(this.config.initialLoadDelay);

},

// add extra information of current weather
// windDirection, humidity, sunrise and sunset
addExtraInfoWeather: function(wrapper) {

var small = document.createElement("div");
small.className = "normal medium";

var windIcon = document.createElement("span");
windIcon.className = "wi wi-strong-wind dimmed";
small.appendChild(windIcon);

var windSpeed = document.createElement("span");
windSpeed.innerHTML = " " + this.windSpeed;
small.appendChild(windSpeed);

if (this.config.showWindDirection) {
var windDirection = document.createElement("sup");
windDirection.innerHTML = " " + this.translate(this.windDirection);
small.appendChild(windDirection);
}
var spacer = document.createElement("span");
spacer.innerHTML = "&nbsp;";
small.appendChild(spacer);

```

```

if (this.config.showHumidity) {
var humidity = document.createElement("span");
humidity.innerHTML = this.humidity;

var spacer = document.createElement("sup");
spacer.innerHTML = "&nbsp;";

var humidityIcon = document.createElement("sup");
humidityIcon.className = "wi wi-humidity humidityIcon";
humidityIcon.innerHTML = "&nbsp;";

small.appendChild(humidity);
small.appendChild(spacer);
small.appendChild(humidityIcon);
}

var sunriseSunsetIcon = document.createElement("span");
sunriseSunsetIcon.className = "wi dimmed " + this.sunriseSunsetIcon;
small.appendChild(sunriseSunsetIcon);

var sunriseSunsetTime = document.createElement("span");
sunriseSunsetTime.innerHTML = " " + this.sunriseSunsetTime;
small.appendChild(sunriseSunsetTime);

wrapper.appendChild(small);
},

// Override dom generator.
getDom: function() {
var wrapper = document.createElement("div");

if (this.config.appid === "") {
wrapper.innerHTML = "Please set the correct openweather <i>appid</i> in the
config for module: " + this.name + ".";
wrapper.className = "dimmed light small";
return wrapper;
}

if (!this.loaded) {
wrapper.innerHTML = this.translate("LOADING");
wrapper.className = "dimmed light small";
return wrapper;
}

if (this.config.onlyTemp === false) {
this.addExtraInfoWeather(wrapper);
}

var large = document.createElement("div");
large.className = "large light";

var weatherIcon = document.createElement("span");
weatherIcon.className = "wi weathericon " + this.weatherType;
large.appendChild(weatherIcon);

```

```

var degreeLabel = "";
if (this.config.degreeLabel) {
switch (this.config.units ) {
case "metric":
degreeLabel = "C";
break;
case "imperial":
degreeLabel = "F";
break;
case "default":
degreeLabel = "K";
break;
}
}

var temperature = document.createElement("span");
temperature.className = "bright";
temperature.innerHTML = " " + this.temperature + "&deg;" + degreeLabel;
large.appendChild(temperature);

wrapper.appendChild(large);
return wrapper;
},

// Override getHeader method.
getHeader: function() {
if (this.config.appendLocationNameToHeader) {
return this.data.header + " " + this.fetchedLocationName;
}

return this.data.header;
},

// Override notification handler.
notificationReceived: function(notification, payload, sender) {
if (notification === "DOM_OBJECTS_CREATED") {
if (this.config.appendLocationNameToHeader) {
this.hide(0, {lockString: this.identifier});
}
}
if (notification === "CALENDAR_EVENTS") {
var senderClasses = sender.data.classes.toLowerCase().split(" ");
if (senderClasses.indexOf(this.config.calendarClass.toLowerCase()) !== -1) {
var lastEvent = this.firstEvent;
this.firstEvent = false;

for (e in payload) {
var event = payload[e];
if (event.location || event.geo) {
this.firstEvent = event;
//Log.log("First upcoming event with location: ", event);
break;
}
}
}
}
}

```

```

    },

    /* updateWeather(compliments)
    * Requests new data from openweather.org.
    * Calls processWeather on succesfull response.
    */
    updateWeather: function() {
        if (this.config.appid === "") {
            Log.error("CurrentWeather: APPID not set!");
            return;
        }

        var url = this.config.apiBase + this.config.apiVersion + "/" +
            this.config.weatherEndpoint + this.getParams();
        var self = this;
        var retry = true;

        var weatherRequest = new XMLHttpRequest();
        weatherRequest.open("GET", url, true);
        weatherRequest.onreadystatechange = function() {
            if (this.readyState === 4) {
                if (this.status === 200) {
                    self.processWeather(JSON.parse(this.response));
                } else if (this.status === 401) {
                    self.updateDom(self.config.animationSpeed);

                    Log.error(self.name + ": Incorrect APPID.");
                    retry = true;
                } else {
                    Log.error(self.name + ": Could not load weather.");
                }
            }
        };

        if (retry) {
            self.scheduleUpdate((self.loaded) ? -1 : self.config.retryDelay);
        }
    };

    weatherRequest.send();
},

    /* getParams(compliments)
    * Generates an url with api parameters based on the config.
    *
    * return String - URL params.
    */
    getParams: function() {
        var params = "?";
        if(this.config.locationID) {
            params += "id=" + this.config.locationID;
        } else if(this.config.location) {
            params += "q=" + this.config.location;
        } else if (this.firstEvent && this.firstEvent.geo) {
            params += "lat=" + this.firstEvent.geo.lat + "&lon=" + this.firstEvent.geo.lon
        } else if (this.firstEvent && this.firstEvent.location) {

```

```

params += "q=" + this.firstEvent.location;
} else {
this.hide(this.config.animationSpeed, {lockString:this.identifier});
return;
}

params += "&units=" + this.config.units;
params += "&lang=" + this.config.lang;
params += "&APPID=" + this.config.appid;

return params;
},

/* processWeather(data)
 * Uses the received data to set the various values.
 *
 * argument data object - Weather information received form openweather.org.
 */
processWeather: function(data) {

if (!data || !data.main || typeof data.main.temp === "undefined") {
// Did not receive usable new data.
// Maybe this needs a better check?
return;
}

this.humidity = parseFloat(data.main.humidity);
this.temperature = this.roundValue(data.main.temp);

if (this.config.useBeaufort){
this.windSpeed = this.ms2Beaufort(this.roundValue(data.wind.speed));
}else {
this.windSpeed = parseFloat(data.wind.speed).toFixed(0);
}

this.windDirection = this.deg2Cardinal(data.wind.deg);
this.weatherType = this.config.iconTable[data.weather[0].icon];

var now = new Date();
var sunrise = new Date(data.sys.sunrise * 1000);
var sunset = new Date(data.sys.sunset * 1000);

// The moment().format('h') method has a bug on the Raspberry Pi.
// So we need to generate the timestring manually.
// See issue: https://github.com/MichMich/MagicMirror/issues/181
var sunriseSunsetDateObject = (sunrise < now && sunset > now) ? sunset :
sunrise;
var timeString = moment(sunriseSunsetDateObject).format("HH:mm");
if (this.config.timeFormat !== 24) {
//var hours = sunriseSunsetDateObject.getHours() % 12 || 12;
if (this.config.showPeriod) {
if (this.config.showPeriodUpper) {
//timeString = hours + moment(sunriseSunsetDateObject).format(':mm A');
timeString = moment(sunriseSunsetDateObject).format("h:mm A");
}
}
}
}

```

```

    } else {
    //timeString = hours + moment(sunriseSunsetDateObject).format(':mm a');
    timeString = moment(sunriseSunsetDateObject).format("h:mm a");
    }
    } else {
    //timeString = hours + moment(sunriseSunsetDateObject).format(':mm');
    timeString = moment(sunriseSunsetDateObject).format("h:mm");
    }
    }

    this.sunriseSunsetTime = timeString;
    this.sunriseSunsetIcon = (sunrise < now && sunset > now) ? "wi-sunset" : "wi-
sunrise";

    this.show(this.config.animationSpeed, {lockString:this.identifier});
    this.loaded = true;
    this.updateDom(this.config.animationSpeed);
    this.sendNotification("CURRENTWEATHER_DATA", {data: data});
    },

    /* scheduleUpdate()
    * Schedule next update.
    *
    * argument delay number - Milliseconds before next update. If empty,
    this.config.updateInterval is used.
    */
    scheduleUpdate: function(delay) {
    var nextLoad = this.config.updateInterval;
    if (typeof delay !== "undefined" && delay >= 0) {
    nextLoad = delay;
    }

    var self = this;
    setTimeout(function() {
    self.updateWeather();
    }, nextLoad);
    },

    /* ms2Beaufort(ms)
    * Converts m2 to beaufort (windspeed).
    *
    * argument ms number - Windspeed in m/s.
    *
    * return number - Windspeed in beaufort.
    */
    ms2Beaufort: function(ms) {
    var kmh = ms * 60 * 60 / 1000;
    var speeds = [1, 5, 11, 19, 28, 38, 49, 61, 74, 88, 102, 117, 1000];
    for (var beaufort in speeds) {
    var speed = speeds[beaufort];
    if (speed > kmh) {
    return beaufort;
    }
    }
    }

```

```

return 12;
},

deg2Cardinal: function(deg) {
if (deg>11.25 && deg<=33.75){
return "NNE";
} else if (deg > 33.75 && deg <= 56.25) {
return "NE";
} else if (deg > 56.25 && deg <= 78.75) {
return "ENE";
} else if (deg > 78.75 && deg <= 101.25) {
return "E";
} else if (deg > 101.25 && deg <= 123.75) {
return "ESE";
} else if (deg > 123.75 && deg <= 146.25) {
return "SE";
} else if (deg > 146.25 && deg <= 168.75) {
return "SSE";
} else if (deg > 168.75 && deg <= 191.25) {
return "S";
} else if (deg > 191.25 && deg <= 213.75) {
return "SSW";
} else if (deg > 213.75 && deg <= 236.25) {
return "SW";
} else if (deg > 236.25 && deg <= 258.75) {
return "WSW";
} else if (deg > 258.75 && deg <= 281.25) {
return "W";
} else if (deg > 281.25 && deg <= 303.75) {
return "WNW";
} else if (deg > 303.75 && deg <= 326.25) {
return "NW";
} else if (deg > 326.25 && deg <= 348.75) {
return "NNW";
} else {
return "N";
}
},

/* function(temperature)
* Rounds a temperature to 1 decimal or integer (depending on config.roundTemp).
*
* argument temperature number - Temperature.
*
* return number - Rounded Temperature.
*/
roundValue: function(temperature) {
var decimals = this.config.roundTemp ? 0 : 1;
return parseFloat(temperature).toFixed(decimals);
}
});

```

To use this module, add it to the modules array in the `config/config.js` file:

```

modules: [
  {
    module: "currentweather",
    position: "top_right",    // This can be any of the regions.
                                // Best results in left or
right regions.
    config: {
      // See 'Configuration options' for more information.
      location: "Amsterdam,Netherlands",
      locationID: "", //Location ID from
http://openweathermap.org/help/city_list.txt
      appid: "abcde12345abcde12345abcde12345ab" //openweathermap.org API
key.
    }
  }
]

```

To edit this file:

<https://github.com/MichMich/MagicMirror/tree/master/modules/default/currentweather>

## Weather Forecast

```

Module.register("weatherforecast",{

  // Default module config.
  defaults: {
    location: false,
    locationID: false,
    appid: "",
    units: config.units,
    maxNumberOfDays: 7,
    showRainAmount: false,
    updateInterval: 10 * 60 * 1000, // every 10 minutes
    animationSpeed: 1000,
    timeFormat: config.timeFormat,
    lang: config.language,
    fade: true,
    fadePoint: 0.25, // Start on 1/4th of the list.
    colored: false,

    initialLoadDelay: 2500, // 2.5 seconds delay. This delay is used to keep the
OpenWeather API happy.
    retryDelay: 2500,

    apiVersion: "2.5",
    apiBase: "http://api.openweathermap.org/data/",
    forecastEndpoint: "forecast/daily",

    appendLocationNameToHeader: true,
    calendarClass: "calendar",

```



```

roundTemp: false,

iconTable: {
"01d": "wi-day-sunny",
"02d": "wi-day-cloudy",
"03d": "wi-cloudy",
"04d": "wi-cloudy-windy",
"09d": "wi-showers",
"10d": "wi-rain",
"11d": "wi-thunderstorm",
"13d": "wi-snow",
"50d": "wi-fog",
"01n": "wi-night-clear",
"02n": "wi-night-cloudy",
"03n": "wi-night-cloudy",
"04n": "wi-night-cloudy",
"09n": "wi-night-showers",
"10n": "wi-night-rain",
"11n": "wi-night-thunderstorm",
"13n": "wi-night-snow",
"50n": "wi-night-alt-cloudy-windy"
},
},

// create a variable for the first upcoming calendar event. Used if no location
is specified.
firstEvent: false,

// create a variable to hold the location name based on the API result.
fetchedLocationName: "",

// Define required scripts.
getScripts: function() {
return ["moment.js"];
},

// Define required styles.
getStyles: function() {
return ["weather-icons.css", "weatherforecast.css"];
},

// Define required translations.
getTranslations: function() {
// The translations for the default modules are defined in the core translation
files.
// Therefore we can just return false. Otherwise we should have returned a
dictionary.
// If you're trying to build your own module including translations, check out
the documentation.
return false;
},

// Define start sequence.
start: function() {
Log.info("Starting module: " + this.name);

```

```

// Set locale.
moment.locale(config.language);

this.forecast = [];
this.loaded = false;
this.scheduleUpdate(this.config.initialLoadDelay);

this.updateTimer = null;

},

// Override dom generator.
getDom: function() {
var wrapper = document.createElement("div");

if (this.config.appid === "") {
wrapper.innerHTML = "Please set the correct openweather <i>appid</i> in the
config for module: " + this.name + ".";
wrapper.className = "dimmed light small";
return wrapper;
}

if (!this.loaded) {
wrapper.innerHTML = this.translate("LOADING");
wrapper.className = "dimmed light small";
return wrapper;
}

var table = document.createElement("table");
table.className = "small";

for (var f in this.forecast) {
var forecast = this.forecast[f];

var row = document.createElement("tr");
if (this.config.colored) {
row.className = "colored";
}
table.appendChild(row);

var dayCell = document.createElement("td");
dayCell.className = "day";
dayCell.innerHTML = forecast.day;
row.appendChild(dayCell);

var iconCell = document.createElement("td");
iconCell.className = "bright weather-icon";
row.appendChild(iconCell);

var icon = document.createElement("span");
icon.className = "wi weathericon " + forecast.icon;
iconCell.appendChild(icon);

var maxTempCell = document.createElement("td");

```

```

maxTempCell.innerHTML = forecast.maxTemp;
maxTempCell.className = "align-right bright max-temp";
row.appendChild(maxTempCell);

var minTempCell = document.createElement("td");
minTempCell.innerHTML = forecast.minTemp;
minTempCell.className = "align-right min-temp";
row.appendChild(minTempCell);

if (this.config.showRainAmount) {
var rainCell = document.createElement("td");
if (isNaN(forecast.rain)) {
rainCell.innerHTML = "";
} else {
if(config.units !== "imperial") {
rainCell.innerHTML = forecast.rain + " mm";
} else {
rainCell.innerHTML = (parseFloat(forecast.rain) / 25.4).toFixed(2) + " in";
}
}
rainCell.className = "align-right bright rain";
row.appendChild(rainCell);
}

if (this.config.fade && this.config.fadePoint < 1) {
if (this.config.fadePoint < 0) {
this.config.fadePoint = 0;
}
var startingPoint = this.forecast.length * this.config.fadePoint;
var steps = this.forecast.length - startingPoint;
if (f >= startingPoint) {
var currentStep = f - startingPoint;
row.style.opacity = 1 - (1 / steps * currentStep);
}
}

return table;
},

// Override getHeader method.
getHeader: function() {
if (this.config.appendLocationNameToHeader) {
return this.data.header + " " + this.fetchedLocationName;
}

return this.data.header;
},

// Override notification handler.
notificationReceived: function(notification, payload, sender) {
if (notification === "DOM_OBJECTS_CREATED") {
if (this.config.appendLocationNameToHeader) {
this.hide(0, {lockString: this.identifier});
}
}
}

```

```

    }
    }
    if (notification === "CALENDAR_EVENTS") {
    var senderClasses = sender.data.classes.toLowerCase().split(" ");
    if (senderClasses.indexOf(this.config.calendarClass.toLowerCase()) !== -1) {
    var lastEvent = this.firstEvent;
    this.firstEvent = false;

    for (e in payload) {
    var event = payload[e];
    if (event.location || event.geo) {
    this.firstEvent = event;
    //Log.log("First upcoming event with location: ", event);
    break;
    }
    }
    }
    },

    /* updateWeather(compliments)
    * Requests new data from openweather.org.
    * Calls processWeather on succesfull response.
    */
    updateWeather: function() {
    if (this.config.appid === "") {
    Log.error("WeatherForecast: APPID not set!");
    return;
    }

    var url = this.config.apiBase + this.config.apiVersion + "/" +
    this.config.forecastEndpoint + this.getParams();
    var self = this;
    var retry = true;

    var weatherRequest = new XMLHttpRequest();
    weatherRequest.open("GET", url, true);
    weatherRequest.onreadystatechange = function() {
    if (this.readyState === 4) {
    if (this.status === 200) {
    self.processWeather(JSON.parse(this.response));
    } else if (this.status === 401) {
    self.updateDom(self.config.animationSpeed);

    Log.error(self.name + ": Incorrect APPID.");
    retry = true;
    } else {
    Log.error(self.name + ": Could not load weather.");
    }
    }

    if (retry) {
    self.scheduleUpdate((self.loaded) ? -1 : self.config.retryDelay);
    }
    }
    };

```

```

weatherRequest.send();
},

/* getParams(compliments)
 * Generates an url with api parameters based on the config.
 *
 * return String - URL params.
 */
getParams: function() {
var params = "?";
if(this.config.locationID) {
params += "id=" + this.config.locationID;
} else if(this.config.location) {
params += "q=" + this.config.location;
} else if (this.firstEvent && this.firstEvent.geo) {
params += "lat=" + this.firstEvent.geo.lat + "&lon=" + this.firstEvent.geo.lon
} else if (this.firstEvent && this.firstEvent.location) {
params += "q=" + this.firstEvent.location;
} else {
this.hide(this.config.animationSpeed, {lockString:this.identifier});
return;
}

params += "&units=" + this.config.units;
params += "&lang=" + this.config.lang;
/*
 * Submit a specific number of days to forecast, between 1 to 16 days.
 * The OpenWeatherMap API properly handles values outside of the 1 - 16 range and
returns 7 days by default.
 * This is simply being pedantic and doing it ourselves.
 */
params += "&cnt=" + (((this.config.maxNumberOfDays < 1) ||
(this.config.maxNumberOfDays > 16)) ? 7 : this.config.maxNumberOfDays);
params += "&APPID=" + this.config.appid;

return params;
},

/* processWeather(data)
 * Uses the received data to set the various values.
 *
 * argument data object - Weather information received form openweather.org.
 */
processWeather: function(data) {
this.fetchedLocatioName = data.city.name + ", " + data.city.country;

this.forecast = [];
for (var i = 0, count = data.list.length; i < count; i++) {

var forecast = data.list[i];
this.forecast.push({

day: moment(forecast.dt, "X").format("ddd"),
icon: this.config.iconTable[forecast.weather[0].icon],
maxTemp: this.roundValue(forecast.temp.max),

```

```

minTemp: this.roundValue(forecast.temp.min),
rain: this.roundValue(forecast.rain)

});
}

//Log.log(this.forecast);
this.show(this.config.animationSpeed, {lockString:this.identifier});
this.loaded = true;
this.updateDom(this.config.animationSpeed);
},

/* scheduleUpdate()
* Schedule next update.
*
* argument delay number - Milliseconds before next update. If empty,
this.config.updateInterval is used.
*/
scheduleUpdate: function(delay) {
var nextLoad = this.config.updateInterval;
if (typeof delay !== "undefined" && delay >= 0) {
nextLoad = delay;
}

var self = this;
clearTimeout(this.updateTimer);
this.updateTimer = setTimeout(function() {
self.updateWeather();
}, nextLoad);
},

/* ms2Beaufort(ms)
* Converts m2 to beaufort (windspeed).
*
* argument ms number - Windspeed in m/s.
*
* return number - Windspeed in beaufort.
*/
ms2Beaufort: function(ms) {
var kmh = ms * 60 * 60 / 1000;
var speeds = [1, 5, 11, 19, 28, 38, 49, 61, 74, 88, 102, 117, 1000];
for (var beaufort in speeds) {
var speed = speeds[beaufort];
if (speed > kmh) {
return beaufort;
}
}
return 12;
},

/* function(temperature)
* Rounds a temperature to 1 decimal or integer (depending on config.roundTemp).
*
* argument temperature number - Temperature.
*

```

```

* return number - Rounded Temperature.
*/
roundValue: function(temperature) {
var decimals = this.config.roundTemp ? 0 : 1;
return parseFloat(temperature).toFixed(decimals);
}
});

```

To use this module, add it to the modules array in the config/config.js file:

```

modules: [
  {
    module: "weatherforecast",
    position: "top_right", // This can be any of the regions. // Best
    results in left or right regions.
    config: {
      // See 'Configuration options' for more information.
      location: "Amsterdam,Netherlands",
      locationID: "", //Location ID from
      http://openweathermap.org/help/city_list.txt
      appid: "abcde12345abcde12345abcde12345ab"
      //openweathermap.org API key.
    }
  }
]

```

To edit this file:

<https://github.com/MichMich/MagicMirror/tree/master/modules/default/weatherforecast>

## Weather Forecast

```

Module.register("newsfeed",{

// Default module config.
defaults: {
feeds: [
  {
title: "New York Times",
url: "http://www.nytimes.com/services/xml/rss/nyt/HomePage.xml",
encoding: "UTF-8" //ISO-8859-1
  }
],
showSourceTitle: true,
showPublishDate: true,
showDescription: false,
wrapTitle: true,
wrapDescription: true,
hideLoading: false,
reloadInterval: 5 * 60 * 1000, // every 5 minutes
updateInterval: 10 * 1000,
animationSpeed: 2.5 * 1000,
maxNewsItems: 0, // 0 for unlimited

```

```

ignoreOldItems: false,
ignoreOlderThan: 24 * 60 * 60 * 1000, // 1 day
removeStartTags: "",
removeEndTags: "",
startTags: [],
endTags: []

},

// Define required scripts.
getScripts: function() {
return ["moment.js"];
},

// Define required translations.
getTranslations: function() {
// The translations for the default modules are defined in the core translation
files.
// Therefor we can just return false. Otherwise we should have returned a
dictionary.
// If you're trying to build your own module including translations, check out
the documentation.
return false;
},

// Define start sequence.
start: function() {
Log.info("Starting module: " + this.name);

// Set locale.
moment.locale(config.language);

this.newsItems = [];
this.loaded = false;
this.activeItem = 0;

this.registerFeeds();

},

// Override socket notification handler.
socketNotificationReceived: function(notification, payload) {
if (notification === "NEWS_ITEMS") {
this.generateFeed(payload);

if (!this.loaded) {
this.scheduleUpdateInterval();
}

this.loaded = true;
}
},

// Override dom generator.
getDom: function() {

```



```

var wrapper = document.createElement("div");

if (this.config.feedUrl) {
wrapper.className = "small bright";
wrapper.innerHTML = "The configuration options for the newsfeed module have
changed.<br>Please check the documentation.";
return wrapper;
}

if (this.activeItem >= this.newsItems.length) {
this.activeItem = 0;
}

if (this.newsItems.length > 0) {

// this.config.showFullArticle is a run-time configuration, triggered by optional
notifications
if (!this.config.showFullArticle && (this.config.showSourceTitle ||
this.config.showPublishDate)) {
var sourceAndTimestamp = document.createElement("div");
sourceAndTimestamp.className = "light small dimmed";

if (this.config.showSourceTitle && this.newsItems[this.activeItem].sourceTitle
!= "") {
sourceAndTimestamp.innerHTML = this.newsItems[this.activeItem].sourceTitle;
}
if (this.config.showSourceTitle && this.newsItems[this.activeItem].sourceTitle
!= "" && this.config.showPublishDate) {
sourceAndTimestamp.innerHTML += ", ";
}
if (this.config.showPublishDate) {
sourceAndTimestamp.innerHTML += moment(new
Date(this.newsItems[this.activeItem].pubdate)).fromNow();
}
if (this.config.showSourceTitle && this.newsItems[this.activeItem].sourceTitle
!= "" || this.config.showPublishDate) {
sourceAndTimestamp.innerHTML += ":";
}

wrapper.appendChild(sourceAndTimestamp);
}

//Remove selected tags from the beginning of rss feed items (title or
description)

if (this.config.removeStartTags == "title" || this.config.removeStartTags ==
"both") {

for (f=0; f<this.config.startTags.length;f++) {
if
(this.newsItems[this.activeItem].title.slice(0,this.config.startTags[f].length)
== this.config.startTags[f]) {
this.newsItems[this.activeItem].title =
this.newsItems[this.activeItem].title.slice(this.config.startTags[f].length,this.
newsItems[this.activeItem].title.length);
}
}
}

```

```

}
}

}

if (this.config.removeStartTags == "description" || this.config.removeStartTags
== "both") {

if (this.config.showDescription) {
for (f=0; f<this.config.startTags.length;f++) {
if
(this.newsItems[this.activeItem].description.slice(0,this.config.startTags[f].len
gth) == this.config.startTags[f]) {
this.newsItems[this.activeItem].title =
this.newsItems[this.activeItem].description.slice(this.config.startTags[f].length
,this.newsItems[this.activeItem].description.length);
}
}
}

}

//Remove selected tags from the end of rss feed items (title or description)

if (this.config.removeEndTags) {
for (f=0; f<this.config.endTags.length;f++) {
if (this.newsItems[this.activeItem].title.slice(-
this.config.endTags[f].length)==this.config.endTags[f]) {
this.newsItems[this.activeItem].title =
this.newsItems[this.activeItem].title.slice(0,-this.config.endTags[f].length);
}
}
}

if (this.config.showDescription) {
for (f=0; f<this.config.endTags.length;f++) {
if (this.newsItems[this.activeItem].description.slice(-
this.config.endTags[f].length)==this.config.endTags[f]) {
this.newsItems[this.activeItem].description =
this.newsItems[this.activeItem].description.slice(0,-
this.config.endTags[f].length);
}
}
}

}

if(!this.config.showFullArticle){
var title = document.createElement("div");
title.className = "bright medium light" + (!this.config.wrapTitle ? " no-wrap" :
"");
title.innerHTML = this.newsItems[this.activeItem].title;
wrapper.appendChild(title);
}

if (this.config.showDescription) {

```

```

var description = document.createElement("div");
description.className = "small light" + (!this.config.wrapDescription ? " no-
wrap" : "");
description.innerHTML = this.newsItems[this.activeItem].description;
wrapper.appendChild(description);
}

if (this.config.showFullArticle) {
var fullArticle = document.createElement("iframe");
fullArticle.className = "";
fullArticle.style.width = "100%";
fullArticle.style.top = "0";
fullArticle.style.left = "0";
fullArticle.style.position = "fixed";
fullArticle.height = window.innerHeight;
fullArticle.style.border = "none";
fullArticle.src = this.newsItems[this.activeItem].url;
wrapper.appendChild(fullArticle);
}

if (this.config.hideLoading) {
this.show();
}

} else {
if (this.config.hideLoading) {
this.hide();
} else {
wrapper.innerHTML = this.translate("LOADING");
wrapper.className = "small dimmed";
}
}

return wrapper;
},

/* registerFeeds()
 * registers the feeds to be used by the backend.
 */
registerFeeds: function() {
for (var f in this.config.feeds) {
var feed = this.config.feeds[f];
this.sendSocketNotification("ADD_FEED", {
feed: feed,
config: this.config
});
}
},

/* generateFeed()
 * Generate an ordered list of items for this configured module.
 *
 * attribute feeds object - An object with feeds returned by the node helper.
 */
generateFeed: function(feeds) {

```

```

var newsItems = [];
for (var feed in feeds) {
var feedItems = feeds[feed];
if (this.subscribedToFeed(feed)) {
for (var i in feedItems) {
var item = feedItems[i];
item.sourceTitle = this.titleForFeed(feed);
if (!(this.config.ignoreOldItems && ((Date.now() - new Date(item.pubdate)) >
this.config.ignoreOlderThan))) {
newsItems.push(item);
}
}
}
}
newsItems.sort(function(a,b) {
var dateA = new Date(a.pubdate);
var dateB = new Date(b.pubdate);
return dateB - dateA;
});
if(this.config.maxNewsItems > 0) {
newsItems = newsItems.slice(0, this.config.maxNewsItems);
}
this.newsItems = newsItems;
},

/* subscribedToFeed(feedUrl)
* Check if this module is configured to show this feed.
*
* attribute feedUrl string - Url of the feed to check.
*
* returns bool
*/
subscribedToFeed: function(feedUrl) {
for (var f in this.config.feeds) {
var feed = this.config.feeds[f];
if (feed.url === feedUrl) {
return true;
}
}
return false;
},

/* titleForFeed(feedUrl)
* Returns title for a specific feed Url.
*
* attribute feedUrl string - Url of the feed to check.
*
* returns string
*/
titleForFeed: function(feedUrl) {
for (var f in this.config.feeds) {
var feed = this.config.feeds[f];
if (feed.url === feedUrl) {
return feed.title || "";
}
}
}

```

```

}
return "";
},

/* scheduleUpdateInterval()
 * Schedule visual update.
 */
scheduleUpdateInterval: function() {
var self = this;

self.updateDom(self.config.animationSpeed);

timer = setInterval(function() {
self.activeItem++;
self.updateDom(self.config.animationSpeed);
}, this.config.updateInterval);
},

/* capitalizeFirstLetter(string)
 * Capitalizes the first character of a string.
 *
 * argument string string - Input string.
 *
 * return string - Capitalized output string.
 */
capitalizeFirstLetter: function(string) {
return string.charAt(0).toUpperCase() + string.slice(1);
},

resetDescrOrFullArticleAndTimer: function() {
this.config.showDescription = false;
this.config.showFullArticle = false;
if(!timer){
this.scheduleUpdateInterval();
}
},

notificationReceived: function(notification, payload, sender) {
Log.info(this.name + " - received notification: " + notification);
if(notification == "ARTICLE_NEXT"){
var before = this.activeItem;
this.activeItem++;
if (this.activeItem >= this.newsItems.length) {
this.activeItem = 0;
}
this.resetDescrOrFullArticleAndTimer();
Log.info(this.name + " - going from article #" + before + " to #" +
this.activeItem + " (of " + this.newsItems.length + ")");
this.updateDom(100);
} else if(notification == "ARTICLE_PREVIOUS"){
var before = this.activeItem;
this.activeItem--;
if (this.activeItem < 0) {
this.activeItem = this.newsItems.length - 1;
}
}
}

```

```

this.resetDescrOrFullArticleAndTimer();
Log.info(this.name + " - going from article #" + before + " to #" +
this.activeItem + " (of " + this.newsItems.length + ")");
this.updateDom(100);
}
// if "more details" is received the first time: show article summary, on second
time show full article
else if(notification == "ARTICLE_MORE_DETAILS"){
this.config.showDescription = !this.config.showDescription;
this.config.showFullArticle = !this.config.showDescription;
clearInterval(timer);
timer = null;
Log.info(this.name + " - showing " + this.config.showDescription ? "article
description" : "full article");
this.updateDom(100);
} else if(notification == "ARTICLE_LESS_DETAILS"){
this.resetDescrOrFullArticleAndTimer();
Log.info(this.name + " - showing only article titles again");
this.updateDom(100);
} else {
Log.info(this.name + " - unknown notification, ignoring: " + notification);
}
},
});

```

To use this module, add it to the modules array in the `config/config.js` file:

```

modules: [
  {
    module: "newsfeed",
    position: "bottom_bar", // This can be any of the regions. Best
results in center regions.
    config: {
      // The config property is optional.
      // If no config is set, an example calendar is shown.
      // See 'Configuration options' for more information.

      feeds: [
        {
          title: "New York Times",
          url:
"http://www.nytimes.com/services/xml/rss/nyt/HomePage.xml",
        },
        {
          title: "BBC",
          url:
"http://feeds.bbci.co.uk/news/video_and_audio/news_front_page/rss.xml?edition=uk",
        },
      ]
    }
  }
]

```

To edit this file:

<https://github.com/MichMich/MagicMirror/tree/master/modules/default/newsfeed>

## Compliments

```
Module.register("compliments", {

  // Module config defaults.
  defaults: {
    compliments: {
      anytime: [
        "Hey there sexy!"
      ],
      morning: [
        "Good morning, handsome!",
        "Enjoy your day!",
        "How was your sleep?"
      ],
      afternoon: [
        "Hello, beauty!",
        "You look sexy!",
        "Looking good today!"
      ],
      evening: [
        "Wow, you look hot!",
        "You look nice!",
        "Hi, sexy!"
      ]
    },
    updateInterval: 30000,
    remoteFile: null,
    fadeSpeed: 4000
  },

  // Set currentweather from module
  currentWeatherType: "",

  // Define required scripts.
  getScripts: function() {
    return ["moment.js"];
  },
```

```

// Define start sequence.
start: function() {
Log.info("Starting module: " + this.name);

this.lastComplimentIndex = -1;

if (this.config.remoteFile != null) {
this.complimentFile((response) => {
this.config.compliments = JSON.parse(response);
});
}

// Schedule update timer.
var self = this;
setInterval(function() {
self.updateDom(self.config.fadeSpeed);
}, this.config.updateInterval);
},

/* randomIndex(compliments)
* Generate a random index for a list of compliments.
*
* argument compliments Array<String> - Array with compliments.
*
* return Number - Random index.
*/
randomIndex: function(compliments) {
if (compliments.length === 1) {
return 0;
}

var generate = function() {
return Math.floor(Math.random() * compliments.length);
};

var complimentIndex = generate();

while (complimentIndex === this.lastComplimentIndex) {
complimentIndex = generate();
}

this.lastComplimentIndex = complimentIndex;

```



```

return complimentIndex;
},

/* complimentArray()
 * Retrieve an array of compliments for the time of the day.
 *
 * return compliments Array<String> - Array with compliments for the time of the
day.
 */
complimentArray: function() {
var hour = moment().hour();
var compliments = null;

if (hour >= 3 && hour < 12) {
compliments = this.config.compliments.morning;
} else if (hour >= 12 && hour < 17) {
compliments = this.config.compliments.afternoon;
} else {
compliments = this.config.compliments.evening;
}

if (typeof compliments === "undefined") {
compliments = new Array();
}

if (this.currentWeatherType in this.config.compliments) {
compliments.push.apply(compliments,
this.config.compliments[this.currentWeatherType]);
}

compliments.push.apply(compliments, this.config.compliments.anytime);

return compliments;
},

/* complimentFile(callback)
 * Retrieve a file from the local filesystem
 */
complimentFile: function(callback) {
var xobj = new XMLHttpRequest();
xobj.overrideMimeType("application/json");

```

```

xobj.open("GET", this.file(this.config.remoteFile), true);
xobj.onreadystatechange = function() {
if (xobj.readyState == 4 && xobj.status == "200") {
callback(xobj.responseText);
}
};
xobj.send(null);
},

/* complimentArray()
* Retrieve a random compliment.
*
* return compliment string - A compliment.
*/
randomCompliment: function() {
var compliments = this.complimentArray();
var index = this.randomIndex(compliments);

return compliments[index];
},

// Override dom generator.
getDom: function() {
var complimentText = this.randomCompliment();

var compliment = document.createTextNode(complimentText);
var wrapper = document.createElement("div");
wrapper.className = this.config.classes ? this.config.classes : "thin xlarge
bright";
wrapper.appendChild(compliment);

return wrapper;
},

// From data currentweather set weather type
setCurrentWeatherType: function(data) {
var weatherIconTable = {
"01d": "day_sunny",
"02d": "day_cloudy",
"03d": "cloudy",
"04d": "cloudy_windy",
"09d": "showers",

```

```

"10d": "rain",
"11d": "thunderstorm",
"13d": "snow",
"50d": "fog",
"01n": "night_clear",
"02n": "night_cloudy",
"03n": "night_cloudy",
"04n": "night_cloudy",
"09n": "night_showers",
"10n": "night_rain",
"11n": "night_thunderstorm",
"13n": "night_snow",
"50n": "night_alt_cloudy_windy"
};
this.currentWeatherType = weatherIconTable[data.weather[0].icon];
},

// Override notification handler.
notificationReceived: function(notification, payload, sender) {
  if (notification == "CURRENTWEATHER_DATA") {
    this.setCurrentWeatherType(payload.data);
  }
},

});

```

To use this module, add it to the modules array in the `config/config.js` file:

```

modules: [
  {
    module: "compliments",
    position: "lower_third", // This can be any of the regions.
                                                                    // Best results in one of
the middle regions like: lower_third
    config: {
      // The config property is optional.
      // If no config is set, an example calendar is shown.
      // See 'Configuration options' for more information.
    }
  }
]

```

To edit this file:

<https://github.com/MichMich/MagicMirror/tree/master/modules/default/compliments>

It is also important to note that this is the section our team edited to make the Smart Mirror more user friendly to the Atwater Kent Community. This is also where we encouraged users to email the MQP team if they would like to be involved with designing modules.

## **Additional Modules**

### **Instagram**

The Instagram feature was added to feature WPI's Instagram feed. How the team did this is can be found within the MQP report. Additional API keys and password can be found in the operational manual given to the ECE technician, Bill Appleyard and Professor Mazumder.

1. Navigate into your MagicMirror's modules folder and execute git clone <https://github.com/kapsolas/MMM-Instagram.git>.
2. A new folder will appear, navigate into it.
3. Execute npm install to install the node dependencies.

To use this module, add it to the modules array in the `config/config.js` file:

```
{
  module: 'MMM-Instagram',
  position: 'top_right',
  config: {
    access_token: 'API_KEY from instagram',
    count: 200,
    min_timestamp: 0,
    animationSpeed: 2500,
    updateInterval: 12500
  }
},
```

### **Code for MMM-Instagram:**

```
Module.register
('MMM-
Instagram', {

  defaults: {
    format: 'json',
    lang: 'en-us',
    id: '',
    animationSpeed: 1000,
    updateInterval: 60000, // 10 minutes
    access_token: '',
    count: 200,
    min_timestamp: 0,
```

```

loadingText: 'Loading...'
},

// Define required scripts
getScripts: function() {
return ["moment.js"];
},

/*
// Define required translations
getTranslations: function() {
return false;
},
*/

// Define start sequence
start: function() {
Log.info('Starting module: ' + this.name);
this.data.classes = 'bright medium';
this.loaded = false;
this.images = {};
this.activeItem = 0;
this.url = 'https://api.instagram.com/v1/users/self/media/recent'
+ this.getParams();
this.grabPhotos();
},

grabPhotos: function() {
// the notifications are not working for some reason... so we
won't do anything asynchronously
// we will just make the call to the method to get the object
with photo links....
//Log.info('sending socket notification: INSTAGRAM_GET and URL: '
+ this.url);
this.sendSocketNotification("INSTAGRAM_GET", this.url);

// this may not be needed... need to think about it.
//setTimeout(this.grabPhotos, this.config.interval, this);
},

getStyles: function() {
return ['instagram.css', 'font-awesome.css'];
}

```

```

},

// Override the dom generator
getDom: function() {
var wrapper = document.createElement("div");
var imageDisplay = document.createElement('div'); //support for
config.changeColor

if (!this.loaded) {
wrapper.innerHTML = this.config.loadingText;
return wrapper;
}

// set the first item in the list...
if (this.activeItem >= this.images.photo.length) {
this.activeItem = 0;
}

var tempimage = this.images.photo[this.activeItem];

// image
var imageLink = document.createElement('div');
//imageLink.innerHTML = "<img
src='https://www.google.com/images/branding/googlelogo/1x/googlel
ogo_color_272x92dp.png'>";
imageLink.id = "MMM-Instagram-image";
imageLink.innerHTML = "<img src='" + tempimage.photolink + "'>";

imageDisplay.appendChild(imageLink);
wrapper.appendChild(imageDisplay);

return wrapper;
},

/* scheduleUpdateInterval()
* Schedule visual update.
*/
scheduleUpdateInterval: function() {
var self = this;

Log.info("Scheduled update interval set up...");
self.updateDom(self.config.animationSpeed);

```

```

setInterval(function() {
Log.info("incrementing the activeItem and refreshing");
self.activeItem++;
self.updateDom(self.config.animationSpeed);
}, this.config.updateInterval);
},

/*
* getParams()
* returns the query string required for the request to flickr to
get the
* photo stream of the user requested
*/
getParams: function() {
var params = '?';
params += 'count=' + this.config.count;
params += '&min_timestamp=' + this.config.min_timestamp;
params += '&access_token=' + this.config.access_token;
return params;
},

// override socketNotificationReceived
socketNotificationReceived: function(notification, payload) {
//Log.info('socketNotificationReceived: ' + notification);
if (notification === 'INSTAGRAM_IMAGE_LIST')
{
//Log.info('received INSTAGRAM_IMAGE_LIST');
this.images = payload;

//Log.info("count: " + this.images.photo.length);

// we want to update the dom the first time and then schedule
next updates
if (!this.loaded) {
this.updateDom(1000);
this.scheduleUpdateInterval();
}

this.loaded = true;
}
}

});

```

## 6.9 Code from LED Panels

### 6.9.1 Power Panel Section

```
PowerPanel  AK_History  AK_MQP_Menu  AKNews  CampusEvents  Pong  SolarData  TopMenu  keyPressed  ▼
1  /*****AK POWER PANEL*****/
2  ***Jonas Ciemy, Alejandro Miranda, Kelsey Powderly
3  ****2015-2016*****/
4
5  ***Bernardino Garay, Clara Merino, Raphael Sarkar, Sonam Sherpa
6  ****2016-2017*****/
7  //LIBRARY INITIALIZATIONS=====
8  import processing.serial.*;
9  import java.util.Calendar;
10 import java.util.Date;
11 import java.util.TimeZone;
12 import java.util.concurrent.ThreadLocalRandom;
13 //import com.onformative.yahooweather.*; //Weather Library from Yahoo
14 import org.dishevelled.processing.executor.Executor;
15 import java.util.concurrent.TimeUnit;
16
17 //String Variables for solar panel current and voltage-----
18 String Sp_Current;
19 String Sp_Voltage;
20 Serial myPort;//Creates an instance of a serial port
21
22 //Values for setting colors
23 int red = 0;
24 int yellow = 60;
25 int green = 120;
26 int cyan = 180;
27 int blue = 240;
28 int purple = 300;
29
30 //Initialize Fonts
31 PFont font8;
32 PFont font16;
33 PFont numberFont;
34
35 //General Vars
36 int windowHeight = 384;
37 int windowHeight = 192;
38 boolean[][] keys;
39
40 //Menu Vars, Strings Provide name of selections, nums provide number of selections
41 String widgets[][] = {"ABOUT", "CAMPUS EVENTS", "AK MAP", "AK HISTORY"}, {"GAMES", "SOLAR DATA", "AK MQP", "AK NEWS"};
42 String games[][] = {"PONG", "OTHER GAMES", "EMAIL"}, {"GOT ANY", "TO ADD?"}, {"akpowerpanels@wp1.edu"};
43 String MQP[][] = {"GAUSS GENERATOR", "FPGA STOPLIGHT"}, {"FIREFIGHTER GPS", "ROTATION SENSOR"};
44 int numWidgets = widgets[0].length;
45
46 int numGames = games[0].length;
47 int numMQP = MQP[0].length;
48 //Positions of the selections
49 int GUIpositionX = 1;
50 int GUIpositionY = 1;
51 int currentWidget = 0;
52 int menuColors[] = new int[2*numWidgets];
53 //Height determines the number of selections on each side of menu
54 int itemHeight = 192/numWidgets;
55 int gameItemHeight = 192/numGames;
56 int MQPItemHeight = 192/numMQP;
57 int menuColorChange = 0;
58
59 //Time and Date Vars
60 String hourAM;
61 String hourPM;
62 String min;
63 String sec;
64 String months[] = {"January", "February", "March", "April", "May", "June", "July", "August", "September", "November", "December"};
65 String days[] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"};
66 Calendar c = Calendar.getInstance(); //Initialize Calendar
67 int timeColors[] = new int[3];
68
69 //RSS Vars
70 String campusURL = "https://api.orgsync.com/api/v3/portals/29444/events.rss?key=dxLpV3wbG6wDsb2fUmu5ImQhUQh_Cs8HP1_CH9gs&upcoming=true&categories%5B%5D=82308&categories%5B%5D=82308&categories%5B%5D=82308";
71 String AKURL = "https://orgsync.com/135668/news_posts/feed";
72 int numberEvents;
73 int numberPosts;
74 XML campusRSS; //Create XML object
75 XML AKRSS;
76 Executor executor;
77 int RSSUpdateInterval = 300; // update every 5 minutes
78 boolean updateRSS1;
79 boolean updateRSS2;
80
81 //Weather Vars
82 //YahooWeather weatherF; //Create weather object
83 int updateIntervalMillis = 300000; //update every 5 minutes
84 PImage[] weatherIcon;
85
86 //Solar Data Vars
87 float[] readings = new float[windowWidth]; // initialized to zeroes
88 int plotWidth = windowWidth;
89 int plotHeight = windowHeight - 16;
```



```

29 int yAxisDivs = 6;
30 int maxYAxis = 100;
31 String[] yAxisLabels = new String[yAxisDivs];
32 String VCPLabels[] = {"Voltage:", "Current:", "Power:"};
33 float VCPValues[] = {0, 0, 0};
34
35 //Load campus maps
36 PImage AK1, AK2, AK3, AKB;
37 int floor=1;
38
39 //Pong Vars
100 int rightPos = 96;
101 int leftPos = 96;
102 int speed = 1;
103 int xball = 192;
104 int yball = 96;
105 int xdirection = 1;
106 int ydirection = 1;
107 int hits = 0;
108 int paddleWidth = 10;
109 int paddleSpeed = 2;
110
111 //Load Images =====
112 //Pizza Friday Images
113 PImage Pizza;
114 //AK MQP Images
115 PImage GaussianCover;
116 PImage GaussianDescription;
117 PImage FPGACover;
118 PImage FPGADescription;
119 PImage FirefighterCover;
120 PImage FirefighterDescription;
121 PImage FiberCover;
122 PImage FiberDescription;
123 //AK History Images
124 PImage AKHistory2;
125 PImage Trolley;
126
127 void setup() {
128   surface.setLocation(-1, 300); //SET THE INITIAL LOCATION OF THE DISPLAYS
129   executor = new Executor(this, 1);
130
131   size(384, 192); // Set up the window
132   noSmooth(); //Turn off smoothing
133
134   colorMode(HSB, 360, 100, 100);
135   //Load fonts=====
136   font8 = loadFont("pixelmfix-8.vlw");
137   font16 = loadFont("pixelmfix-16.vlw");
138   numberFont = loadFont("visitorMono.vlw");
139   noStroke();
140
141   //Menu Colors=====
142   for (int i=0; i<(numWidgets*2); i++) {
143     menuColors[1] = i*(360/(numWidgets*2));
144   }
145   executor.repeat("menuColor", 0, 1000, TimeUnit.MILLISECONDS);
146
147   //Time Colors=====
148   for (int i=0; i<3; i++) {
149     timeColors[1] = i*120;
150   }
151   executor.repeat("timeColor", 0, 1000, TimeUnit.MILLISECONDS);
152
153   //SolarPanel=====
154   // printArray(Serial.list());
155   myPort = new Serial(this, Serial.list()[2], 9600); //Preparing Communication between arduino and Processing via serial communication. '2' is the specific port
156
157   for (int i=0; i<yAxisDivs; i++) {
158     yAxisLabels[i] = String.valueOf(maxYAxis/yAxisDivs*i);
159   }
160
161   //Loading AK Maps=====
162   AK1 = loadImage("AK1.png");
163   AK2 = loadImage("AK2.png");
164   AK3 = loadImage("AK3.png");
165   AKB = loadImage("AKB.png");
166
167   //Loading AK MQP=====
168   GaussianCover = loadImage("Gaussian Cover.PNG");
169   GaussianCover.resize(384, 192);
170   GaussianDescription = loadImage("Gaussian Description.PNG");
171   GaussianDescription.resize(384, 192);
172   FPGACover = loadImage("FPGA Cover.PNG");
173   FPGACover.resize(384, 192);
174   FPGADescription = loadImage("FPGA Description.PNG");
175   FPGADescription.resize(384, 192);
176   FirefighterCover = loadImage("Firefighter Cover.PNG");
177   FirefighterCover.resize(384, 192);
178   FirefighterDescription = loadImage("Firefighter Description.PNG");

```

```

177 FirefighterDescription.resize(384, 192);
178 FiberCover = loadImage("Fiber Cover.PNG");
179 FiberCover.resize(384, 192);
180 FiberDescription = loadImage("Fiber Description.PNG");
181 FiberDescription.resize(384, 192);
182 //Loading AK History=====
183 AKHistory2 = loadImage("AKHistory2.PNG");
184 AKHistory2.resize(384, 192);
185 Trolley = loadImage("Trolley.png");
186 Trolley.resize(384, 192);
187
188 //RSS=====
189 executor.repeat("RSSupdate1", 0, RSSupdateInterval, TimeUnit.SECONDS);
190 executor.repeat("RSSupdate2", 0, RSSupdateInterval, TimeUnit.SECONDS);
191
192 //Pong=====
193 keys = new boolean[2][9];
194 for (int i=1; i<9; i++) {
195     keys[0][i] = false;
196     keys[1][i] = false;
197 }
198
199 //PIZZA FRIDAY IMAGES =====
200 Pizza = loadImage("Images/pizza.png");
201 Pizza.resize(64, 0);
202 }
203 // Cases with (1-99) represent top menu, (100-199) represent menu for games, (200-299) represent menu for AK History
204 void draw() {
205     frame.toFront();
206     frame.requestFocus();
207     background(0);
208
209     if (keys[0][2]||keys[1][2])
210         currentWidget = 0; //go to menu
211
212     switch(currentWidget) {
213     case 0:
214         topMenu();
215         break;
216     case 1:
217         AKpower();
218         break;
219     case 2:
220         RSS();
221         break;
222     case 3:
223         AKmap();
224         break;
225     case 4:
226         Hhistory();
227         break;
228     case 5:
229         gameMenu();
230         break;
231     case 6:
232         solarData();
233         break;
234     case 7:
235         MQPMenu();
236         break;
237     case 8:
238         AKnews();
239         break;
240
241     case 101:
242         pong();
243         break;
244
245     case 201:
246         Gauss();
247         break;
248     case 202:
249         Gauss2();
250         break;
251     case 203:
252         Gauss3();
253         break;
254     case 204:
255         Gauss4();
256         break;
257     }
258 }
259 }

```

## AK History Section

```
PowerPanel AK_History AK_MQP_Menu AKnews CampusEvents Pong SolarData TopMenu keyPressed
1 //AK History slides. Currently has two pictures and descriptions for each one
2 void History(){
3
4     textAlign(CENTER);
5
6
7     if (keys[0][1]|keys[1][1] & (floor < 1)) {
8         floor++;
9         keys[0][1] = false;
10        keys[1][1] = false;
11    }
12    if (keys[0][7]|keys[1][7] & (floor > 0)) {
13        floor--;
14        keys[0][7] = false;
15        keys[1][7] = false;
16    }
17
18    switch(floor){
19    case 0:
20    image(AKHistory2,0,0);
21        break;
22    case 1:
23    image(Trolley,0,0);
24        break;
25    }
26 }
```

## AK MQP Menu Section

```
PowerPanel AK_History AK_MQP_Menu AKnews CampusEvents Pong SolarData TopMenu keyPressed
1 //Allows users to navigate through the AK MQP menu, similar to the program in the Top Menu tab
2 void MQPMenu(){
3
4     textFont(font16);
5     textAlign(CENTER);
6     rectMode(CENTER);
7     for (int i=0; i<numMQP; i++){
8         strokeWeight(0);
9         fill(menuColors[2+i],100,20);
10        stroke(menuColors[2+i],100,20);
11        rect(384/4, MQPItemHeight+i*MQPItemHeight/2, 192,MQPItemHeight);
12        fill(menuColors[2+i],100,100);
13        text(MQP[0][i], 384/4, MQPItemHeight+i*MQPItemHeight/2+2);
14
15        fill(menuColors[2+i+1],100,20);
16        stroke(menuColors[2+i+1],100,20);
17        rect(384*3/4, MQPItemHeight+i*MQPItemHeight/2, 192,MQPItemHeight);
18        fill(menuColors[2+i+1],100,100);
19        text(MQP[1][i], 384*3/4, MQPItemHeight+i*MQPItemHeight/2+2);
20    }
21    if ((keys[0][3]|keys[1][3]) & (GUIpositionX == 2))
22        GUIpositionX = 1;
23    if ((keys[0][5]|keys[1][5]) & (GUIpositionX == 1))
24        GUIpositionX = 2;
25    if ((keys[0][1]|keys[1][1]) & (GUIpositionY > 1)){
26        GUIpositionY--;
27        keys[0][1] = false;
28        keys[1][1] = false;
29    }
30    if ((keys[0][7]|keys[1][7]) & (GUIpositionY < numMQP)){
31        GUIpositionY++;
32        keys[0][7] = false;
33        keys[1][7] = false;
34    }
35    if ((keys[0][4]|keys[1][4]) & keyPressed == true){
36        currentWidget = (GUIpositionX-1)*numMQP+GUIpositionY+200;
37        GUIpositionX = 1;
38        GUIpositionY = 1;
39    }
40    noFill();
41    strokeWeight(4);
42    stroke(0);
43    rect(384*(GUIpositionX+2-1)/4, MQPItemHeight*(GUIpositionY-1)+MQPItemHeight/2, 188, MQPItemHeight-4); //rect(centerX, centerY, width, height)
44 }
```

## AK News Section

```
PowerPanel | AKnews | About_Navigation | GameMenu | MQPs | Map | Pong | SolarData | TopMenu | keyPressed | keyReleased ▼
1 //Allows users to post messages onto this sections to be displayed
2 void AKnews(){
3
4     textAlign(LEFT);
5
6     if (updateRSS1 == true){
7         AKRSS = loadXML(AKURL); //load rss feed from url
8         updateRSS1 = false;
9     }
10
11     XML[] eventTitle = AKRSS.getChildren("channel/item/title");
12     XML[] eventDesc = AKRSS.getChildren("channel/item/description");
13     numberPosts = eventTitle.length;
14     String[] titles = new String[numberPosts];
15     String[] descriptions = new String[numberPosts];
16
17     for (int i = 0; i < numberPosts; i++) {
18         titles[i] = eventTitle[i].getContent();
19         descriptions[i] = eventDesc[i].getContent();
20     }
21
22     for (int i = 0; i < numberPosts; i++){ //Print everything out
23         textFont(font16);
24         fill(200,100,100);
25         text(titles[i], 2, 2*(i+1)*16-20);
26         textFont(font8);
27         fill(360);
28         text(descriptions[i].replaceAll("<p>","").replaceAll("</p>\n", ""), 2, 2*(i+1)*16-4);
29     }
30 }
31
32 void RSSupdate1(){
33     println("RSS Update");
34     updateRSS1 = true;
35 }
```

## About Navigation Section

```
PowerPanel | AK_History | AK_MQP_Menu | AKnews | About_Navigation | CampusEvents | TopMenu | keyPressed | keyReleased ▼
1 // About slides displaying three slides for users to navigate through as slides provide instructions and details for each selection
2 void AKpower(){
3     if (keys[0][1]|keys[1][1] & (floor < 2)) {
4         floor++;
5         keys[0][1] = false;
6         keys[1][1] = false;
7     }
8     if (keys[0][7]|keys[1][7] & (floor > 0)) {
9         floor--;
10        keys[0][7] = false;
11        keys[1][7] = false;
12    }
13
14    switch(floor){
15
16    case 0:
17        textFont(numberFont, 30);
18        fill(360);
19        text("3/3", 350,175);
20        String Menu[] = {"AK MAP", "AK MQP", "AK HISTORY", "AK NEWS"};
21        String PizzaPrice[] = {"LAYOUT OF AK BUILDING", "ON ALL FLOORS", "MAJOR QUALIFYING PROJECTS", "WITHIN ECE DEPARTMENT", "HISTORY OF AK", "AND ITS SIGNIFICANCE", "DISPLAY POSTS"};
22        textAlign(LEFT);
23        textFont(font16);
24        textSize(13);
25
26        for (int x = 0; x <= Menu.length - 1; x++) { //LOOP THROUGH THE PIZZA MENU
27            fill(100 + x*20, 100 - x*10, 100);
28            text(Menu[x], 0, (192/Menu.length)*x + 15);
29        }
30        for (int x = 0; x < PizzaPrice.length; x++){ //LOOP THROUGH THE PRICE MENU
31            fill(200 + x*20, 100 + x*10, 100);
32            text(PizzaPrice[x], 95, (192/PizzaPrice.length)*x + 15 );
33        }
34        break;
35
36    case 1:
37        textFont(numberFont, 30);
38        fill(360);
39        text("2/3", 350,175);
40        String Billy[] = {"PUSH", "AND", "TO", "THROUGH"};
41        String Allen[] = {"UP", "DOWN", "NAVIGATE", "MENU"};
42        textAlign(LEFT);
43        textFont(font16);
44        textSize(20);
```

```

46 for (int x = 0; x <= Billy.length - 1; x++) { //LOOP THROUGH THE PIZZA MENU
47     fill(100 + x*20, 100 - x*10, 100);
48     text(Billy[x], 60, (192/Billy.length)*x + 15);
49 }
50 for (int x = 0; x < Allen.length; x++){ //LOOP THROUGH THE PRICE MENU
51     fill(100 + x*20, 100 + x*10, 100);
52     text(Allen[x], 185, (192/Allen.length)*x + 15 );
53 }
54 break;
55
56 case 2:
57     textFont(numberFont, 30);
58     fill(360);
59     text("1/3", 350, 175);
60     String About[] = {"CAMPUS EVENTS", "GAMES", "SOLAR DATA"};
61     String Price[] = {"ANNOUNCEMENTS FOR IMPORTANT", "CAMPUS EVENTS", "PLAY GAMES USING", "TOUCH SESNORS", "DISPLAYS DATA FROM", "SOLAR PANELS ON ROOF"};
62     textAlign(LEFT);
63     textFont(font16);
64     textSize(13);
65
66 for (int x = 0; x <= About.length - 1; x++) { //LOOP THROUGH THE PIZZA MENU
67     fill(100 + x*20, 100 - x*10, 100);
68     text(About[x], 0, (192/About.length)*x + 15);
69 }
70 for (int x = 0; x < Price.length; x++){ //LOOP THROUGH THE PRICE MENU
71     fill(200 + x*20, 100 + x*10, 100);
72     text(Price[x], 125, (192/Price.length)*x + 15 );
73 }
74 break;
75 }
76 }

```

## Campus Event Section

PowerPanel	AK_History	AK_MQP_Menu	AKnews	About_Navigation	CampusEvents						TopMenu	keyPressed	
------------	------------	-------------	--------	------------------	--------------	--	--	--	--	--	---------	------------	--

```

1 //Information is gathered from WPI to be displayed as campus events
2 void RSS(){
3
4     textAlign(LEFT);
5
6     if (updateRSS2 == true){
7         campusRSS = loadXML(campusURL); //load rss feed from url
8         updateRSS2 = false;
9     }
10
11     XML[] dateStart = campusRSS.getChildren("channel/item/event:startdate"); //Create array of all start dates-times
12     numberEvents = dateStart.length; //determine number of events
13     String[] dayOfWeekRSS = new String[numberEvents];
14     String[] timeStart = new String[numberEvents];
15     for (int i = 0; i < numberEvents; i++) {
16         String start = dateStart[i].getContent();
17         String[] startSplit = start.split("[-T:]");
18         c.set(Integer.parseInt(startSplit[0]), Integer.parseInt(startSplit[1]), Integer.parseInt(startSplit[2]));
19         int day = c.get(Calendar.DAY_OF_WEEK);
20         dayOfWeekRSS[i] = days[day-1];
21
22         //Time
23         if (Integer.parseInt(startSplit[3])+1 > 12){
24             timeStart[i] = (Integer.parseInt(startSplit[3])+1-12)+":"+startSplit[4];
25         }else{
26             timeStart[i] = (Integer.parseInt(startSplit[3])+1)+":"+startSplit[4];
27         }
28     }
29
30     XML[] dateEnd = campusRSS.getChildren("channel/item/event:enddate");
31     String[] timeEnd = new String[numberEvents];
32     for (int i = 0; i < numberEvents; i++) {
33         String end = dateEnd[i].getContent();
34         String[] endSplit = end.split("[-T:]");
35
36         //Time
37         if (Integer.parseInt(endSplit[3])+1 > 12)
38             timeEnd[i] = (Integer.parseInt(endSplit[3])+1-12)+":"+endSplit[4];
39         else
40             timeEnd[i] = (Integer.parseInt(endSplit[3])+1)+":"+endSplit[4];
41     }
42
43     XML[] locations = campusRSS.getChildren("channel/item/event:location");
44     String[] location = new String[numberEvents];

```

```

45 for (int i = 0; i < numberEvents; i++) {
46     location[i] = locations[i].getContent();
47 }
48
49 XML[] titles = campusRSS.getChildren("channel/item/title");
50 String[] name = new String[numberEvents];
51 for (int i = 0; i < numberEvents; i++) {
52     name[i] = titles[i].getContent();
53 }
54
55 for (int i = 0; i < numberEvents; i++){ //Print everything out
56     textFont(font16);
57     fill(200,100,100);
58     text(name[i], 2, 2*(i+1)*16-20);
59     textFont(font8);
60     fill(360);
61     text(dayOfWeekRSS[i]+" "+timeStart[i]+"-"+timeEnd[i]+" "+location[i], 2, 2*(i+1)*16-4);
62 }
63 }
64
65 void RSSupdate2(){
66     println("RSS Update");
67     updateRSS2 = true;
68 }

```

## Game Menu Section

PowerPanel	AK_History	AK_MQP_Menu	AKnews	About_Navigation	CampusEvents	GameMenu	keyPressed
------------	------------	-------------	--------	------------------	--------------	----------	------------

```

1 //Similar to Top Menu except provides options to select games
2 void gameMenu(){
3     textFont(font16);
4     textAlign(CENTER);
5     rectMode(CENTER);
6     for (int i=0; i<numGames; i++){
7         strokeWeight(0);
8         fill(menuColors[2+i],100,20);
9         stroke(menuColors[2+i],100,20);
10        rect(384/4, gameItemHeight+i*gameItemHeight/2, 192,gameItemHeight);
11        fill(menuColors[2+i],100,100);
12        text(games[0][i], 384/4, gameItemHeight+i*gameItemHeight/2+2);
13
14        fill(menuColors[2+i+1],100,20);
15        stroke(menuColors[2+i+1],100,20);
16        rect(384*3/4, gameItemHeight+i*gameItemHeight/2, 192,gameItemHeight);
17        fill(menuColors[2+i+1],100,100);
18        text(games[1][i], 384*3/4, gameItemHeight+i*gameItemHeight/2+2);
19    }
20
21    if ((keys[0][3]|keys[1][3]) & (GUIpositionX == 2))
22        GUIpositionX = 1;
23    if ((keys[0][5]|keys[1][5]) & (GUIpositionX == 1))
24        GUIpositionX = 2;
25    if ((keys[0][1]|keys[1][1]) & (GUIpositionY > 1)){
26        GUIpositionY--;
27        keys[0][1] = false;
28        keys[1][1] = false;
29    }
30    if ((keys[0][7]|keys[1][7]) & (GUIpositionY < numGames)){
31        GUIpositionY++;
32        keys[0][7] = false;
33        keys[1][7] = false;
34    }
35    if ((keys[0][4]|keys[1][4]) & keyPressed == true){
36        currentWidget = (GUIpositionX-1)*numGames+GUIpositionY+100;
37        GUIpositionX = 1;
38        GUIpositionY = 1;
39    }
40    noFill();
41    strokeWeight(4);
42    stroke(0);
43    rect(384*(GUIpositionX+1)/4, gameItemHeight+(GUIpositionY-1)*gameItemHeight/2, 188, gameItemHeight-4); //rect(centerX, centerY, width, height)
44 }

```

## MQPs Section

```
PowerPanel  AK_History  AK_MQP_Menu  AKnews  About_Navigation  CampusEvents  GameMenu  MQPs  keyPressed
1 //Slides for each set of the four available MQP's to review
2 void Gauss(){
3
4     textAlign(CENTER);
5
6
7     if (keys[0][1]|keys[1][1] & (floor < 1)) {
8         floor++;
9         keys[0][1] = false;
10        keys[1][1] = false;
11    }
12    if (keys[0][7]|keys[1][7] & (floor > 0)) {
13        floor--;
14        keys[0][7] = false;
15        keys[1][7] = false;
16    }
17
18    switch(floor){
19    case 0:
20        image(GaussianDescription,0,0);
21        textFont(numberFont, 30);
22        fill(360);
23        text("2/2", 370,190);
24        break;
25    case 1:
26        image(GaussianCover,0,5);
27        textFont(numberFont, 30);
28        fill(360);
29        text("1/2", 370,190);
30        break;
31    }
32 }
33 void Gauss2(){
34
35     textAlign(CENTER);
36
37
38     if (keys[0][1]|keys[1][1] & (floor < 1)) {
39         floor++;
40         keys[0][1] = false;
41         keys[1][1] = false;
42     }
43     if (keys[0][7]|keys[1][7] & (floor > 0)) {
44         floor--;
```

```
PowerPanel  AK_History  AK_MQP_Menu  AKnews  About_Navigation  CampusEvents  GameMenu  MQPs  keyPressed
45     keys[0][7] = false;
46     keys[1][7] = false;
47 }
48
49 switch(floor){
50 case 0:
51     image(FPGADescription,0,0);
52     textFont(numberFont, 30);
53     fill(360);
54     text("2/2", 370,190);
55     break;
56 case 1:
57     image(FPGACover,0,0);
58     textFont(numberFont, 30);
59     fill(360);
60     text("1/2", 370,190);
61     break;
62 }
63 }
64 void Gauss3(){
65
66     textAlign(CENTER);
67
68
69     if (keys[0][1]|keys[1][1] & (floor < 1)) {
70         floor++;
71         keys[0][1] = false;
72         keys[1][1] = false;
73     }
74     if (keys[0][7]|keys[1][7] & (floor > 0)) {
75         floor--;
76         keys[0][7] = false;
77         keys[1][7] = false;
78     }
79
80     switch(floor){
81     case 0:
82         image(FirefighterDescription,0,3);
83         textFont(numberFont, 30);
84         fill(360);
85         text("2/2", 370,190);
86         break;
87     case 1:
88         image(FirefighterCover,0,0);
```





# Pong

```
PowerPanel | AKnews | About_Navigation | GameMenu | MQPs | Map | Pong | SolarData | TopMenu | keyPressed | keyReleased
1 //Program of pong game. Does need some work such as victory, score board, reset option, pause menu, and ball movement
2 void pong(){
3
4   stroke(360);
5   strokeWeight(4);
6   strokeCap(SQUARE);
7
8
9   line(0, leftPos+2*paddleWidth, 0, leftPos-2*paddleWidth);
10  line(380, rightPos+2*paddleWidth, 380, rightPos-2*paddleWidth);
11
12  if (keys[0][7] & (leftPos < 180)) {
13    leftPos = leftPos + paddleSpeed;
14  }
15  if (keys[0][1] & (leftPos > 0)) {
16    leftPos = leftPos - paddleSpeed;
17  }
18
19  if (keys[1][7] & (rightPos < 180)) {
20    rightPos = rightPos + paddleSpeed;
21  }
22  if (keys[1][1] & (rightPos > 0)) {
23    rightPos = rightPos - paddleSpeed;
24  }
25
26  ellipse(xball, yball, 8, 8);
27
28  xball = xball + xdirection*speed;
29  yball = yball + ydirection*speed;
30
31  if (yball > 192)
32    ydirection = -1;
33  if (yball < 0)
34    ydirection = 1;
35  if ((xball > 384) & (rightPos-2*paddleWidth < yball & yball < rightPos+2*paddleWidth)){
36    xdirection = -1;
37    hits++;
38  }
39  if ((xball < 0) & (leftPos-2*paddleWidth < yball & yball < leftPos+2*paddleWidth)){
40    xdirection = 1;
41    hits++;
42  }
43 }
44 }
```

# SolarData Section

```
PowerPanel | AK_MQP_Menu | AKnews | About_Navigation | CampusEvents | GameMenu | MQPs | Map | Pong | SolarData
37 VCPvalues[2] = $pPower;
38 print("The POWER from the Solar Panel is: ");
39 println($pPower);
40 }
41
42 textFont(font8);
43 textAlign(LEFT);
44
45 readings>windowWidth - 1] = ($pPower); //store power to current position
46 readings>windowWidth - 1] = map(readings>windowWidth - 1], 0, maxYAxis, 0, 192);
47
48 // Plot all the points
49 for (int i = 0; i < plotWidth - 1; i++)
50 {
51   if (readings[i] != 0) // Don't plot anything at zero, as that's the default value in the array
52   {
53     stroke(green, 100, 100);
54     line(i, 192 - readings[i]-2, 1,192);
55     stroke(green, 100, 100);
56     line(i, 192 - readings[i], 1, 192);
57   }
58   readings[i] = readings[i + 1]; // Shift the readings to the left so can put the newest reading in
59 }
60
61 for (int i=0; i<3; i++) {
62   fill(100*1, 100, 100);
63   text(VCPLabels[i], 384+i/3+20, 190);
64   text(VCPvalues[i], 384+i/3+VCPLabels[i].length()+7+10, 190);
65 }
66 stroke(360);
67
68 for (int i=0; i<cyAxisDivs; i++) {
69   line(0, 192/6*(i+1), 385, 192/6*(i+1));
70   fill(120/(i+1), 100, 100);
71   text(yAxisLabels[yAxisDivs-i-1]+"W", 1, 192/6*(i+1)-1);
72 }
73 VA = 0;
74 Vsp = 0;
75 }
76
77 float map_f(float x, float in_min, float in_max, float out_min, float out_max)
78 {
79   return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
80 }
```

## TopMenu Section

```
PowerPanel  AKnews  About_Navigation  CampusEvents  GameMenu  MQPs  Map  Pong  SolarData  TopMenu  ▼
1 //Allows users to navigate main menu selections
2 void topMenu(){
3
4   textFont(font16);
5   textAlign(CENTER);
6   rectMode(CENTER);
7
8   for (int i=0; i<numWidgets; i++){
9     strokeWeight(0);
10    fill(menuColors[2+i],100,20);
11    stroke(menuColors[2+i],100,20);
12    rect(384/4, itemHeight+i*itemHeight/2, 192,itemHeight);
13    fill(menuColors[2+i],100,100);
14    text(widgets[0][i], 384/4, itemHeight+i*itemHeight/2+2);
15
16    fill(menuColors[2+i+1],100,20);
17    stroke(menuColors[2+i+1],100,20);
18    rect(384*3/4, itemHeight+i*itemHeight/2, 192,itemHeight);
19    fill(menuColors[2+i+1],100,100);
20    text(widgets[1][i], 384*3/4, itemHeight+i*itemHeight/2+2);
21  }
22
23  if ((keys[0][3]|keys[1][3]) & (GUIpositionX == 2))
24    GUIpositionX = 1;
25  if ((keys[0][5]|keys[1][5]) & (GUIpositionX == 1))
26    GUIpositionX = 2;
27  if ((keys[0][1]|keys[1][1]) & (GUIpositionY > 1)){
28    GUIpositionY--;
29    keys[0][1] = false;
30    keys[1][1] = false;
31  }
32  if ((keys[0][7]|keys[1][7]) & (GUIpositionY < numWidgets)){
33    GUIpositionY++;
34    keys[0][7] = false;
35    keys[1][7] = false;
36  }
37  if (keys[0][4]|keys[1][4]){
38    currentWidget = (GUIpositionX-1)*numWidgets+GUIpositionY;
39    keyPressed = false;
40    GUIpositionX = 1;
41    GUIpositionY = 1;
42  }
43  //Draw Selection Box
44  noFill();
45
46  strokeWeight(4);
47  stroke(0);
48  rect(384*(GUIpositionX+2-1)/4, itemHeight*(GUIpositionY-1)+itemHeight/2, 188, itemHeight-4); //rect(centerX, centerY, width, height)
49  }
50  void menuColor(){
51    for (int i=0; i<(numWidgets*2); i++) {
52      menuColors[i]++;
53      if (menuColors[i] > 361)
54        menuColors[i]=0;
55    }
56  }
```

## KeyPressed Section

```
PowerPanel | ANews | About_Navigation | CampusEvents | GameMenu | MQPs | Map | Pong | SolarData | TopMenu | keyPressed
1 //Detects when keyboard is pressed
2 void keyPressed()
3 {
4   //Pad #1
5   if (key == 'q')
6     keys[0][0] = true;
7   if (key == 'w')
8     keys[0][1] = true;
9   if (key == 'e')
10    keys[0][2] = true;
11  if (key == 'a')
12    keys[0][3] = true;
13  if (key == 's')
14    keys[0][4] = true;
15  if (key == 'd')
16    keys[0][6] = true;
17  if (key == 'z')
18    keys[0][6] = true;
19  if (key == 'x')
20    keys[0][7] = true;
21  if (key == 'c')
22    keys[0][8] = true;
23
24  //Pad #2
25  if (key == 'r')
26    keys[1][0] = true;
27  if (key == 't')
28    keys[1][1] = true;
29  if (key == 'y')
30    keys[1][2] = true;
31  if (key == 'f')
32    keys[1][3] = true;
33  if (key == 'g')
34    keys[1][4] = true;
35  if (key == 'h')
36    keys[1][5] = true;
37  if (key == 'v')
38    keys[1][6] = true;
39  if (key == 'b')
40    keys[1][7] = true;
41  if (key == 'n')
42    keys[1][8] = true;
43 }
```

## KeyReleased Section

```
PowerPanel | CampusEvents | GameMenu | MQPs | Map | Pong | SolarData | TopMenu | keyPressed | keyReleased
1 //Detects when keyboard is released
2 void keyReleased()
3 {
4   //Pad #1
5   if (key == 'q')
6     keys[0][0] = false;
7   if (key == 'w')
8     keys[0][1] = false;
9   if (key == 'e')
10    keys[0][2] = false;
11  if (key == 'a')
12    keys[0][3] = false;
13  if (key == 's')
14    keys[0][4] = false;
15  if (key == 'd')
16    keys[0][6] = false;
17  if (key == 'z')
18    keys[0][6] = false;
19  if (key == 'x')
20    keys[0][7] = false;
21  if (key == 'c')
22    keys[0][8] = false;
23
24  //Pad #2
25  if (key == 'r')
26    keys[1][0] = false;
27  if (key == 't')
28    keys[1][1] = false;
29  if (key == 'y')
30    keys[1][2] = false;
31  if (key == 'f')
32    keys[1][3] = false;
33  if (key == 'g')
34    keys[1][4] = false;
35  if (key == 'h')
36    keys[1][5] = false;
37  if (key == 'v')
38    keys[1][6] = false;
39  if (key == 'b')
40    keys[1][7] = false;
41  if (key == 'n')
42    keys[1][8] = false;
43 }
```

## 6.9.2 Children Panels Section

```
Children_Panels HorizontalTicker Pizza_Friday_cd Weather iFacts iGif1 iGif2 timeDate wpi_twitter_timeline
1 // AK POWER PANEL HQP | TRIAL 1 | 4_14_2016 |
2 // Updated Panel HQP | TRIAL 2 | 4_20_2017 |
3
4 //LIBRARY INITIALIZATIONS=====
5 import processing.serial.*;
6 import java.util.Calendar;
7 import java.util.Date;
8 import java.util.TimeZone;
9
10 //TWITTER INITIALIZATIONS =====
11 //REQUIRES THE TWITTER4J LIBRARY REFER TO http://codasign.com/tutorials/processing-and-twitter/processing-and-twitter-getting-started/ FOR INSTALLING TO PROCESSING API
12 import twitter4j.conf.*;
13 import twitter4j.*;
14 import twitter4j.auth.*;
15 import twitter4j.api.*;
16 import java.util.*;
17
18 //HORIZONTAL TICKER X COORDINATES USED =====
19 int x = 768; //X COORDINATE FOR THE RSS NEW FEED
20 int x1 = 768; //X COORDINATE FOR THE TWITTER FEEDS FOR HORIZONTAL TICKER
21 int x2 = 768; //X COORDINATE FOR THE CONTACT NAZUNDER INFORMATION
22 int f = 0;
23
24 //WEATHER INITIALIZATIONS =====
25 String weatherURL = "http://www.myweather2.com/developer/forecast.aspx?uac=fJc-PDxkpb&output=xml&query=01699&temp_unit=f&ws_unit=mph";
26 XML weather;
27
28 //FONT INITIALIZATIONS =====
29 PFont font8;
30 PFont font16;
31 PFont numberFont;
32
33 //RSS FEED INITIALIZATIONS=====
34 String url = "http://www.techtimes.com/rss/sections/personaltech.xml"; //Current rss feed
35 XML rss;
36
37 //TIMING COUNT VARIABLES =====
38 long Count = 0; // SYSTEM COUNTER
39 int Count_ch1 = 0; // //FACTS
40 int Count_ch2 = 0;
41 int Count_thanks = 0; //COUNT FOR THANK YOU MESSAGE
42 boolean flag = false; //PIZZA FRIDAY PACMAN FLAG
43 boolean flag2 = false; //USED FOR THE HORIZONTAL TICKER
44 boolean flag3 = false; //SPECIAL THANKS FLAG
45
46 boolean flag4 = false; //FOR WEATHER RSS FEED
47
48 //Time and Date Vars
49 String hourAM;
50 String hourPM;
51 String min;
52 String sec;
53 Calendar c = Calendar.getInstance(); //Initialize Calender
54 String days[] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"};
55 String months[] = {"Jan.", "Feb.", "Mar.", "Apr.", "May", "June", "July", "Aug.", "Sep.", "Oct.", "Nov.", "Dec."};
56 int timeColors[] = new int[3];
57
58 //iFacts INITIALIZATIONS =====
59 int Number_of_Fact = 6; //NUMBER OF PROFESSOR PORTRAITS
60 String Facts_Name[] = {"Fact1", "Fact2", "Fact3", "Fact4", "Fact5"};
61 PImage[] Facts; //INSTANTIATE AN ARRAY PROFESSOR PIMAGE OBJECT
62 int fact;
63
64 //PIZZA FRIDAY =====
65 PImage Pizza; //INSTANTIATE A PIMAGE FOR PIZZA IMAGE
66 int numFrames = 3; //NUMBER OF FRAMES REQUIRED FOR PACMAN ANIMATION
67 PImage[] Pacman = new PImage[numFrames]; //INSTANTIATE PACMAN PIMAGE
68
69 //iGif1 =====
70 int no_of_frame = 36;
71 PImage[] gif1;
72 String frame1[] = {"frame_0_delay-0.03s",
73 "frame_1_delay-0.03s",
74 "frame_2_delay-0.03s",
75 "frame_3_delay-0.03s",
76 "frame_4_delay-0.03s",
77 "frame_5_delay-0.03s",
78 "frame_6_delay-0.03s",
79 "frame_7_delay-0.03s",
80 "frame_8_delay-0.03s",
81 "frame_9_delay-0.03s",
82 "frame_10_delay-0.03s",
83 "frame_11_delay-0.03s",
84 "frame_12_delay-0.03s",
85 "frame_13_delay-0.03s",
86 "frame_14_delay-0.03s",
87 "frame_15_delay-0.03s",
88 "frame_16_delay-0.03s",
89 "frame_17_delay-0.03s",
```

```

89         *frame_18_delay-0.03s",
90         *frame_19_delay-0.03s",
91         *frame_20_delay-0.03s",
92         *frame_21_delay-0.03s",
93         *frame_22_delay-0.03s",
94         *frame_23_delay-0.03s",
95         *frame_24_delay-0.03s",
96         *frame_25_delay-0.03s",
97         *frame_26_delay-0.03s",
98         *frame_27_delay-0.03s",
99         *frame_28_delay-0.03s",
100        *frame_29_delay-0.03s",
101        *frame_30_delay-0.03s",
102        *frame_31_delay-0.03s",
103        *frame_32_delay-0.03s",
104        *frame_33_delay-0.03s",
105        *frame_34_delay-0.03s",
106        *frame_35_delay-0.03s",
107        *frame_36_delay-0.03s"
108    };
109
110    // gif2
111    PImage[] gif2;
112    String frame2[] = {"frame_0_delay-0.14s",
113        *frame_1_delay-0.14s",
114        *frame_2_delay-0.14s",
115        *frame_3_delay-0.14s",
116        *frame_4_delay-0.14s",
117        *frame_5_delay-0.14s",
118        *frame_6_delay-0.14s",
119        *frame_7_delay-0.14s",
120        *frame_8_delay-0.14s",
121        *frame_9_delay-0.14s",
122        *frame_10_delay-0.14s",
123        *frame_11_delay-0.14s",
124        *frame_12_delay-0.14s",
125        *frame_13_delay-0.14s",
126        *frame_14_delay-0.14s",
127        *frame_15_delay-0.14s",
128        *frame_16_delay-0.14s",
129        *frame_17_delay-0.14s",
130        *frame_18_delay-0.14s",
131        *frame_19_delay-0.14s",
132        *frame_20_delay-0.14s",

```

```

133
134        *frame_20_delay-0.14s",
135        *frame_21_delay-0.14s",
136        *frame_22_delay-0.14s",
137        *frame_23_delay-0.14s",
138        *frame_24_delay-0.14s",
139        *frame_25_delay-0.14s",
140        *frame_26_delay-0.14s",
141        *frame_27_delay-0.14s",
142        *frame_28_delay-0.14s",
143        *frame_29_delay-0.14s",
144        *frame_30_delay-0.14s",
145        *frame_31_delay-0.14s",
146        *frame_32_delay-0.14s",
147        *frame_33_delay-0.14s",
148        *frame_34_delay-0.14s",
149        *frame_35_delay-0.14s",
150        *frame_36_delay-0.14s"
151    };

```

```

151    //TWITTER INITIALIZATIONS FOR @MPI-----
152    Twitter twitter;
153    List<Status> tweets;
154    int currentTweet;
155    String userName = "MPI"; //USERNAME
156    String Tweet_display;
157    Status status;
158
159    //WEATHER RSS FEEDS -----
160    int Number_of_weather_icons = 95; //NUMBER OF WEATHER ICONS
161    PImage[] weatherIcon = new PImage[Number_of_weather_icons]; //INSTANTIATE WEATHERICON OBJECT
162
163    void setup() {
164        surface.setLocation(-1, 21); //SET THE INITIAL LOCATION OF THE DISPLAYS
165
166        //SETTING UP THE WINDOW-----
167        size(768, 224); // DIMENSIONS OF THE WINDOW
168        noSmooth(); //Turn off smoothing
169        colorMode(HSB, 360, 100, 100); //SET COLORS FOR HSB MODE FOR HUE SATURATION AND BRIGHTNESS
170
171        //LOADING THE IMAGES -----
172
173        //LOAD PIZZA FRIDAY IMAGES-----
174        Pizza = loadImage("Images/pizza.png");
175        Pacman[0] = loadImage("Pacman/pacman1.png");

```

```

176 Pacnan[1] = loadImage("Pacnan/pacnan2.png");
177 Pacnan[2] = loadImage("Pacnan/pacnan3.png");
178 Pacnan[0].resize(35, 35);
179 Pacnan[1].resize(35, 35);
180 Pacnan[2].resize(35, 35);
181
182 //LOAD GIF1 Basketball IMAGES=====
183 gif1 = new PImage[no_of_frame];
184 for (int i = 0; i < gif1.length; i++) {
185     gif1[i] = loadImage("frames/Basketball/"+frame1[i]+".gif");
186 }
187
188 //LOAD GIF2 GopheI IMAGES=====
189 gif2 = new PImage[no_of_frame];
190 for (int i = 0; i < gif2.length; i++) {
191     gif2[i] = loadImage("frames/GopheI/"+frame2[i]+".gif");
192 }
193
194 //LOADING THE FONTS =====
195 font8 = loadFont("pixelmix-8.vlw");
196 font16 = loadFont("pixelmix-16.vlw");
197 numberFont = loadFont("visitorMono.vlw");
198
199 //TWITTER SETUP =====
200 //FOLLOW THIS SAME CONFIGURATION
201 ConfigurationBuilder cb = new ConfigurationBuilder();
202 cb.setAuthConsumerKey("RT0wFQTZRF5THv0I8RUZjF2y8");
203 cb.setAuthConsumerSecret("Lp]lIEE20a0kE16dSYnIFAEyghL6IWL0aeHhRdGd7Hcm7EBs");
204 cb.setAuthAccessToken("728748163183849475-eypLEjNtSM8e1JFkaNkasvQ8zKx8h");
205 cb.setAuthAccessTokenSecret("K51k76P8ZBIX9I1vLj41e1X2hpdwa0nCfXY04MR6Rk3IR");
206 TwitterFactory tf = new TwitterFactory(cb.build());
207 twitter = tf.getInstance();
208 getNewTweets(); //BUFFERS NEW TWEETS
209 currentTweet = 0; //THE CURRENT TWEET BEING DISPLAYED AT A GIVEN TIME
210 thread("refreshTweets"); //CREATE A NEW INSTANCE OF A THREAD
211
212 //LOAD RSS FEEDS =====
213 rss= loadXML(url); //load rss feed
214
215 //LOAD WEATHER XML =====
216 weather = loadXML(weatherURL); //load rss feed from url
217
218 //PREPARE WEATHER ICONS FOR RSSFEED =====
219 for (int i=0; i<95; i++) {
220     for (int i=0; i<95; i++) {
221         weatherIcon[i]=loadImage("WeatherIcons/"+str(i) + ".gif");
222     }
223 }
224
225 void draw() {
226     background(0);
227     strokeWeight(1);
228     stroke(180);
229     noFill();
230
231     //CALL THE VARIOUS MODULES TO DISPLAY WHAT THEY ARE SUPPOSED TO =====
232     flag = pizzaFriday(flag);
233     flag2 = Scrolling_Tech_Feed(flag2);
234     flag4 = weather(flag4);
235     %gif2();
236     %gif1();
237     timeDate();
238     %facts();
239
240     //INCREMENT THE END COUNTER: USED FOR CYCLING THROUGH IMAGES =====
241     Count = Count + 1;
242 }
243
244 //TWITTER FUNCTIONS =====
245 void getNewTweets() {
246     try {
247         tweets = twitter.getUserTimeline(userName);
248     } catch (TwitterException te) {
249         System.out.println("Failed to search tweets: " + te.getMessage());
250         System.exit(-1);
251     }
252 }
253
254 void refreshTweets() { //EVERY 30000 SECONDS REFRESH THE BUFFER WITH NEW TWEETS
255     while (true)
256     {
257         getNewTweets();
258         println("UpdatedTweets");
259         delay(30000);
260     }
261 }

```

## HorizontalTicker Section

```
Children_Panels HorizontalTicker Pizza_Friday_cd Weather iFacts iGif1 iGif2 timeDate wpi_twitter_timeline ▾
1 boolean Scrolling_Tech_Feed(boolean flag) {
2     int TotalLength = 0;
3     frameRate(60); //Determines speed of text; Lower numbers represent slower speeds
4     colorMode(HSB, 360, 100, 100);
5     fill(360); //White Text
6
7     //RSS FEED INFORMATION =====
8     XML[] titleXMLElements = rss.getChildren("channel/item/title"); // Get title of each element
9     XML[] descriptionXMLElements = rss.getChildren("channel/item/description"); // Get description of each element
10    String title = titleXMLElements.toString(); //convert to String for display
11    String description = descriptionXMLElements.toString(); //convert to String for display
12
13    for ( i <= titleXMLElements.length) {
14        title = titleXMLElements[i].getContent(); //get title content from titleXMLElements array
15        description = descriptionXMLElements[i].getContent(); //get description content from descriptionXMLElements array
16        TotalLength = (int)textWidth(title) + (int)textWidth(description); //create TotalLength which is the length of title plus description
17    }
18
19    //TWITTER INFORMATION =====
20    twitter_update();
21    if (flag == false) { //THIS IS FOR LOADING DATA DURING THE FIRST ROUND
22        //TWITTER INFORMATION =====
23        status = tweets.get(currentTweet);
24        Tweet_display = status.getText();
25        x = 650 + Tweet_display.length()*3; //X POSITION FOR THE TWITTER DISPLAY
26        //RSS FEED INFORMATION =====
27        x1 = 650 + TotalLength; //X POSITION FOR THE RSS FEED
28        flag = true;
29    }
30
31    //GET THE NEXT TWEET AFTER CERTAIN AMOUNT OF TIME =====
32    if (x < 0 - status.getText().length()*3) {
33        Status status = tweets.get(currentTweet); //GET THE CURRENT TWEET.
34        Tweet_display = status.getText();
35        x = 650 + Tweet_display.length()*3;
36    }
37    //GET THE NEXT RSS AFTER CERTAIN AMOUNT OF TIME =====
38    if (x1 < 0 - TotalLength) {
39        i++;
40        x1 = 650 + TotalLength;
41    }
42    // Draw the text
43    textFont(font16);
44    fill(140, 100, 100);
45    text(title + '!' + '!' + description, x1, 15); //DRAW THE RSS FEED
46
47    textFont(font8);
48    fill(200, 100, 100);
49    text(Tweet_display, x, 20); //DRAW THE TWITTER MESSAGE FROM @MPI
50
51    //DECREMENT THE HORIZONTAL(X) VALUES TO SCROLL THE TEXT AFTER EACH CLOCK CYCLE =====
52    x--;
53    x1--;
54
55    //If all of the xml elements have been displayed, reset to beginning and display all elements again
56    if ( i > titleXMLElements.length) {
57        i=0;
58    }
59    return flag;
60 }
}
```

## Pizza Friday Countdown Section

```
Children_Panels HorizontalTicker Pizza_Friday_cd Weather iFacts iGif1 iGif2 timeDate wpi_twitter_timeline ▾
1 boolean pizzafriday(boolean flag) {
2     colorMode(RGB);
3     int current_day = (Calendar.getInstance().get(Calendar.DAY_OF_WEEK));
4     int day_remap[] = {0, 2, 3, 4, 5, 6, 7, 1};
5
6     // Interval of 35 clock cycles between pacman images
7     if (Count % 35 == 0) {
8         flag = !flag;
9     }
10    //DISPLAY PACMAN AT THE CORRECT POSITION BASED ON CURRENT DAY
11    if (flag == true) {
12        for (int x = 1; x <= (7 - day_remap[current_day]); x++) {
13            image(Pacman[2], (384 + 255) - (x + day_remap[current_day])*35, 28);
14        }
15        image(Pacman[0], (384 + 255) - day_remap[current_day]*35, 28);
16    } else if (flag == false) {
17        for (int x = 1; x <= (7 - day_remap[current_day]); x++) {
18            image(Pacman[2], (384 + 255) - (x + day_remap[current_day])*35, 28);
19        }
20        image(Pacman[1], (384 + 255) - day_remap[current_day]*35, 28);
21    }
22    image(Pizza, 384, 32); //location of Pizza icon
23    return flag;
24 }
}
```

## Weather Section

```
Children_Panels  HorizontalTicker  Pizza_Friday_cd  Weather  iFacts  iGif1  iGif2  timeDate  wpi_twitter_timeline  ▾
1 boolean weather(boolean flag) {
2
3     if (Count % 10000000 == 0) {          //UPDATE THE WEATHER AFTER EVERY 10000000 CLOCK CYCLES
4         weather = loadXML(weatherURL); //load rss feed from url
5     }
6     //COLLECT THE RIGHT XML DATA FROM THE WEATHER RSS FEED =====
7     XML currentTemp = weather.getChild("current_weather/temp");
8     XML currentWind = weather.getChild("current_weather/wind/speed");
9     XML currentDesc = weather.getChild("current_weather/weather_text");
10    XML currentCode = weather.getChild("current_weather/weather_code");
11    XML [] forecast = weather.getChildren("forecast");
12    XML dayHigh = forecast[0].getChild("day_max_temp");
13    XML dayLow = forecast[0].getChild("night_min_temp");
14    XML dayCode = forecast[0].getChild("day_weather_code");
15    XML tmrwHigh = forecast[1].getChild("day_max_temp");
16    XML tmrwLow = forecast[1].getChild("night_min_temp");
17    XML tmrwCode = forecast[1].getChild("day_weather_code");
18
19    fill(300);
20    textFont(font8);
21    textSize(8);
22    textAlign(LEFT);
23
24    //DISPLAY THE CURRENT WEATHER DATA THAT WAS COLLECTED =====
25    text(currentDesc.getContent(), 768 - 3*32, 104); //Moves description that's above the temperature
26    text(currentTemp.getContent() + "°F", 768 - 3*32, 112); //Moves the temperature
27    text(currentWind.getContent() + " MPH Wind", 768 - 3*32, 120); //Moves description for wind speed
28    image(weatherIcon[Integer.parseInt(currentCode.getContent())], 768 - 4*32, 96, 32, 32); //Moves location of weather image
29    //Display THE CURRENT DAY FORECAST =====
30    fill(0, 100, 100); //Daily High Temperature -Color Red
31    text(dayHigh.getContent(), 768 - 32*3, 140);
32    fill(225, 100, 100); //Daily Low Temperature - Color Blue
33    text(dayLow.getContent(), 768 - 32*3, 148);
34    image(weatherIcon[Integer.parseInt(dayCode.getContent())], 768 - 4*32, 128, 32, 32);
35    //DISPLAY TOMORROW'S DATA (Same as above code but for right side) =====
36    fill(0, 100, 100);
37    text(tmrwHigh.getContent(), 768 - 2*32 + 32, 140);
38    fill(225, 100, 100);
39    text(tmrwLow.getContent(), 768 - 2*32 + 32, 148);
40    image(weatherIcon[Integer.parseInt(tmrwCode.getContent())], 768 - 32*2, 128, 32, 32);
41
42    return flag;
43 }
```



## iFacts Section

```
Children_Panels HorizontalTicker Pizza_Friday_cd Weather iFacts iGif1 iGif2 timeDate wpi_twitter_timeline ▾
1 void iFacts() {
2   // 500 clock cycle between each WPI Facts
3   if (Count % 500 == 0) {
4     Count_ch2 = Count_ch2 + 1;
5     if (Count_ch2 == Number_of_Fact) {
6       Count_ch2 = 0;
7       fact = 0;
8     }
9   }
10  textFont(font8);
11  textAlign(CENTER); //Center text on coordinates
12  fill(360);
13
14  // Count_ch2 represents number of fact being displayed
15  // Using switch case statement to change Facts with Count_ch2 as variable
16  switch(Count_ch2) {
17    case 0:
18      text("WPI", 670, 45);
19      text("Ranking", 670, 55);
20      text("in", 670, 65);
21      text("US", 670, 75);
22      text("Colleges", 670, 85);
23      break;
24    case 1:
25      //display WPI Facts
26      text("#1 Most", 670, 45);
27      text("popular", 670, 55);
28      text("study", 670, 65);
29      text("abroad", 670, 75);
30      text("program", 670, 85);
31      break;
32    case 2:
33      text("#7 College", 670, 45);
34      text("grads with", 670, 55);
35      text("highest", 670, 65);
36      text("earning", 670, 75);
37      text("potential", 670, 85);
38      break;
39    case 3:
40      text("#4 Best", 670, 45);
41      text("College", 670, 55);
42      text("for", 670, 65);
43      text("Greek", 670, 75);
44      text("life", 670, 85);
45
46      break;
47    case 4:
48      text("#17 Best", 670, 45);
49      text("value", 670, 55);
50      text("college", 670, 65);
51      text("return on", 670, 75);
52      text("investment", 670, 85);
53      break;
54    case 5:
55      text("#12",670,45);
56      text("Electrical", 670, 55);
57      text("engineering", 670, 65);
58      text("major in", 670, 75);
59      text("the nation", 670, 85);
60      break;
61  }
62 }
```

## iGif1 Section

```
Children_Panels HorizontalTicker Pizza_Friday_cd Weather iFacts iGif1 iGif2 timeDate wpi_twitter_timeline ▾
1
2 void iGif1() {
3   // 10 clock cycle between each Gif Frame
4   if (Count % 10 == 0) {
5     Count_ch1 = Count_ch1 + 1;
6     if (Count_ch1 == no_of_frame) {
7       Count_ch1 = 0;
8     }
9   }
10  // location of the gif1
11  image(gif1[Count_ch1],575,95,63,63);
12 }
```

## iGif2 Section

```
Children_Panels HorizontalTicker Pizza_Friday_cd Weather iFacts iGif1 iGif2 timeDate wpi_twitter_timeline ▼
1 void iGif2() {
2     // 48 clock cycle between each Gif Frame
3     if (Count % 48 == 0) {
4         Count_ch1 = Count_ch1 + 1;
5         if (Count_ch1 == no_of_frame) {
6             Count_ch1 = 0;
7         }
8     }
9     // location of gif2
10    image(gif2[Count_ch1],702,159,63,63);
11 }
```

## TimeDate Section

```
Children_Panels HorizontalTicker Pizza_Friday_cd Weather iFacts iGif1 iGif2 timeDate wpi_twitter_timeline ▼
1 void timeDate() {
2     int dayOfWeek = c.get(Calendar.DAY_OF_WEEK); //Get current day of week (1-7)
3     textFont(numberFont, 40);
4     textAlign(LEFT);
5
6     //Format all times to 2 digits
7     min = String.format("%02d", minute());
8     hourAM = String.format("%02d", hour());
9     hourPM = String.format("%02d", hour() % 12); //hour is given in 24hr, so subtract 12 to make 12hr
10
11    //Display hours
12    fill(timeColors[0],100,100);
13    if (hour() > 12){ //when hour 13-24, display in 12hr format instead
14        text(hourPM+ ":", 703, 50);
15    }else{
16        text(hourAM+ ":", 703, 50);
17    }
18    fill(timeColors[1],100,100);
19    text(min, 738, 50); //Display minute
20
21    textFont(font8);
22    textAlign(LEFT); //Center text on coordinates
23    fill(360);
24
25    //display DAY, MONTH DATE, YEAR
26    text(days[dayOfWeek-1], 715, 80);
27    text(months[month()-1]+ " " +day(),719, 70);
28    text(year(), 724, 80);
29 }
```

## Wpi twitter timeline Section

```
Children_Panels HorizontalTicker Pizza_Friday_cd Weather iFacts iGif1 iGif2 timeDate wpi_twitter_timeline ▼
1 void twitter_update() { //UPDATES THE TWITTER
2     if (Count % 500 == 0) { //UPDATE THE SCREEN AFTER 500 CYCLES
3         currentTweet = currentTweet + 1; //GET THE NEXT TWEET
4         if (currentTweet >= tweets.size()) //DO NOT COUNT OVER THE SIZE OF TWEETS IN THE BUFFER.
5         {
6             currentTweet = 0;
7         }
8     }
9 }
```

## 7. References

- ODROID | Hardkernel." ODROID. Hardkernel, n.d. Web. 1 Dec. 2016. <[http://www.hardkernel.com/main/products/prdt\\_info.php?g\\_code=G143452239825](http://www.hardkernel.com/main/products/prdt_info.php?g_code=G143452239825)>
- Grid Independent Charging Station with Power Flow Display MQP Report, WPI, 2012
- M. Patkar, "6 best raspberry pi smart mirror projects we've seen so far," in MakeUseOf, 2016. [Online]. Available: <http://www.makeuseof.com/tag/6-best-raspberry-pi-smart-mirror-projects-weve-seen-far/>.
- B. Buy, "BenQ - GL2760H 27" LED HD monitor - glossy black," Best Buy, 2016. [Online]. Available: <http://www.bestbuy.com/site/benq-gl2760h-27-led-hd-monitor-glossy-black/4648606.p?skuId=4648606>.
- "2-Way Mirrored Acrylic Sheets," in Tap Plastics. [Online]. Available: [http://www.tapplastics.com/product/plastics/cut to size plastic/two way mirrored acrylic/558](http://www.tapplastics.com/product/plastics/cut%20to%20size%20plastic/two%20way%20mirrored%20acrylic/558).
- "Raspberry pi 3 - model B - ARMv8 with 1G RAM ID: 3055 - \$39.95: Adafruit industries, unique & fun DIY electronics and kits," in Adafruit. [Online]. Available: <https://www.adafruit.com/product/3055>.
- "SparkFun RGB and Gesture Sensor - APDS-9960," in SparkFun. [Online]. Available: <https://www.sparkfun.com/products/12787>.
- "Mini Push Button Switch," in SparkFun. [Online]. Available: <https://www.sparkfun.com/products/97>.
- WPA supplicant. (n.d.). Retrieved April 26, 2017, from [https://wiki.archlinux.org/index.php/WPA\\_supplicant](https://wiki.archlinux.org/index.php/WPA_supplicant)
- Connect to WPI Wireless using Linux. (n.d.). Retrieved April 26, 2017, from <https://it.wpi.edu/Article/Connect-to-WPI-Wireless-using-Linux>

- Play, W. W. (n.d.). Automatically connect a Raspberry Pi to a Wifi network. Retrieved April 26, 2017, from <http://weworkweplay.com/play/automatically-connect-a-raspberry-pi-to-a-wifi-network/>
- How do I type on the next line in the Terminal? (n.d.). Retrieved April 26, 2017, from <http://askubuntu.com/questions/226204/how-do-i-type-on-the-next-line-in-the-terminal>
- Working with File Permissions on Your Raspberry Pi. (n.d.). Retrieved April 26, 2017, from <http://www.dummies.com/computers/raspberry-pi/working-with-file-permissions-on-your-raspberry-pi/>
- K. (2016, June 15). Complete Setup Tutorial. Retrieved April 26, 2017, from <https://forum.magicmirror.builders/topic/236/complete-setup-tutorial/6>
- How to generate an Instagram Access Token. (n.d.). Retrieved April 26, 2017, from <http://jelled.com/instagram/access-token>
- Ubuntu Documentation. (n.d.). Retrieved April 26, 2017, from <https://help.ubuntu.com/community/Lubuntu/Documentation/CustomizingTheClock>
- “Americanizing” the Raspberry Pi. (2012, April 21). Retrieved April 26, 2017, from <http://rohankapoor.com/2012/04/americanizing-the-raspberry-pi/>
- M. (n.d.). MichMich/MagicMirror. Retrieved April 26, 2017, from <https://github.com/MichMich/MagicMirror/wiki/Auto-Starting-MagicMirror>
- M., L., S., L., T., & X. (2017, April 12). IpWhitelist HowTo. Retrieved April 26, 2017, from <https://forum.magicmirror.builders/topic/1326/ipwhitelist-howto/2>
- K. (2016, July 29). Kapsolas/MMM-Instagram. Retrieved April 26, 2017, from <https://github.com/kapsolas/MMM-Instagram>