A Modularized Approach for Engineering Experimentation Measurements

A Major Qualifying Project Report

Submitted to the faculty

of the

Worchester Polytechnic Institute

By:

Jeffrey Brathwaite

Advisor:

Prof. John Sullivan

Table of Contents

Abstract:

This Major Qualifying Project developed and tested a learning manual composed of separate modules that focus on specific engineering tasks, such as detecting motion, proximity, pressure, control and more. Each module has given tasks to evaluate the student's understanding of the information within each module. The final module is an open ended experiment for the students that uses the knowledge from multiple previous modules to accomplish a multi-step task.

Executive Summary:

As an engineering student, I sought enrollment to WPI for its project-based learning and hands-on experimentation practices and experiences. The facilities of this institute do more than enough to supply any curious minded student with a project with the space and instruments they need to achieve their goal. Unfortunately much of the tools provided require a substantial amount financial investment and an instructor that can guide you on their uses. In this wide world there are far more mechanically minded people out there that do not have access to this school either because of their living arrangement or the lack of funds to attend. I pursued a project using alternative resources that could supply a similar experimental experience in a modularized package with minimal financial investment required by the student.

The fully assembled modules holds the combined lessons I've taken myself and put them into separate lesson plans with the necessary electronic parts. The first module introduces students to the microcontroller being used in this project: the Raspberry Pi. This module will introduce wiring and some slight programming to get used to operating the Raspberry Pi. The first section of this project will properly teach the assembly and use of the Raspberry Pi in conjunction with a basic circuit. Inside this section there is background information and history of the electronics used in this module. The module package includes all the necessary circuit parts and everything the Raspberry Pi needs to operate. At the end of the lesson there will be a short experiment to test the use of the Raspberry Pi along with the given circuits and LED's.

Module 2 will introduce Motors to the students and operating them with the Raspberry Pi. The module teaches the students about 3 different motors: the Stepper, Servo, and DC motors. Each motor's construction and uses are covered in this module to tell the student all the information they would need to understand how the individual motors work. There is also in-

depth instruction on wiring up the Raspberry Pi to the individual motors and basic programming code to start the motors up. This module will also test the students programing and wiring capabilities at the end of it.

Module 3 brings in sensors for data gathering and recording the data inside of Raspberry Pi. This module covers the ultrasonic and the PIR motion sensors, these sensors are used throughout many applications and the student gets to experiment with them. Inside the module there is information on how theses sensors interact with the outside world and how they record data to any given device they are wired to. Both sensors and a mounting plate are supplied with instructions on how to wire the sensors into a circuit that connects and interacts with the Raspberry Pi, along with example code for the student to upload into the Raspberry Pi for testing. Similar to the previous modules the third module has a short experiment for the student to test movements and distances for the Raspberry Pi to record and state if there is a change in movement and distance.

All of these lessons are utilized in the last Module 4 as a final experiment, the student will devise a construction with the Raspberry Pi and the motors/sensors given in the package. There is no explicit lesson in the last module, rather it is an open ended experiment for the student to flex their skills on an aspects of their choosing. The module itself gives options and potential projects for the student to look into but the choice and the method is completely up to them.

Acknowledgements:

I would like to take this time to thank Professor John Sullivan for being patient with me in when my workload would overwhelm me, Mrs. Barbara L. Furhman for being able to obtain the necessary pieces for my project, and of course my dear friends that kept me sane and healthy during this trying time

Introduction:

From the beginning there was an idea of the end product; an affordable, all in one package for prospective students to measure states like temperature and pressure alongside experimenting with control devices and data acquisition. The initial goals were for students to be able to evaluate different factors like temperature and monitor different states; the question became how to get it done. At the start the only direction given was an example course from WPI that teaches students a method of experimenting with different sensors and recording devices. This course utilizes multiple external sensors, a program called LabVIEW[1], and of course a data collecting device simply known as the DAQ (data acquisition) box. Of all the material in use, the DAQ box is the most expensive part of the course; a single multifunction I/O (C series) such as the NI USB-6229 BNC unit costs $3,289 dollars[2] -just to record the data from the various sensors[3]. For this MQP I sought to create a far more affordable experimentation package using similar sensors, instructions for a programming language, and a data recording device to collect the data.

Assembling this kit started off with a difficult choice of determining which data recording device could replicate most the existing DAQ capabilities with an acceptable accuracy and reproducibility. The device of choice to put into this kit is the Raspberry Pi. This microcontroller boasts its own lineup of ready to order sensors that I used in this kit along with some extra motors and sensors for prospective students to pick up and learn. I myself had

[1] Johnson, G. W
[2] National instruments
[3] National instruments

experience with the Raspberry Pi and the programming language Python which made it easier to put together a lesson plan for this project. Within the lesson plan I put together some code and wiring instructions for each component in the kit, and also created a template, for future production, of a base plate that can fit the various motors and sensors so that the student can more easily wire up and test the components as the student sees fit.

The following pages in this report are the individual modules that encompasses the lessons and labs that I have put together for potential students. The first Module encompasses an introduction to assembling and using the Raspberry Pi. Module 1 sets the structure for the rest of the modules by setting up how the Raspberry Pi interacts with other components like wires and motors. Module 2 starts off with Motors, first with a general introduction to basic definitions of motors and how they are used. This is followed by Module 3 which covers various sensors and how to record data using the Raspberry Pi. The final module is the culmination of all of the modules with a final experiment that will combine all of the techniques presented in the report.

Objectives for the Overall Set of Modules:

The purpose of all the modules in this program is to give an alternative method of learning engineering experimentation outside the classroom environment. The standard classroom format is divided between lectures and labs. The lectures are informational sections that deliver engineering theories to students and gives them a chance to ask the teachers questions. The Lab sections of the course focus on trying to put theory to practice. The engineering experimentation laboratories at WPI are equipped with various equipment such as computers, data acquisition devices and sensors to measure, record, and control specific physical environments. The machines are frequently updated and maintained but the cost of buying and using these machines can leave a sizeable debt in many budgets.

An example expense of the machinery in the lab is the Data Acquisition device, or DAQ box as it is called, which can go to four hundred dollars per work station. This also does not take into account the necessary wires that are needed to interface with any given components necessary for the experiments. This gets really expensive when it comes to catering for multiple students at once and having to maintain all of the equipment. This set of modules seeks to provide similar laboratory measurement and control experiences at a more financially viable option for potential students.

To achieve this goal of creating an economically viable option for this experimentation we planned on using a micro-controller to replace the data acquisition device and the computer itself. To combine the two pieces of technology together research was done on some of the latest technology to see if there was a way to both minimize the amount of space used and the amount of money that would have to be spent. Out of all options of integrating a Data acquisition device and a computer together short of buying a specific desktop, we found the most versatile and obtainable answer is the Micro controller.

Traditionally, micro controllers were passion projects for a select few people but they have grown into everyday tools over time. Thanks to the interest and the support from the programming community, the micro controller’s price decreased significantly, allowing for it to be viable for both schools and students.

With the module pack we provide essential sensors and motors to experiment with, an electrical platform called the Breadboard, an integrated H-bridge microchip, and the main microcomputer Raspberry Pi. This module will go into the background and the basic operations and programming of the computer Raspberry Pi as well as an introductory look at the Breadboard and essential wiring introduction to electrical circuits. The Raspberry Pi is miniature computer has the same capabilities of full sized desktops or laptops with the benefits of a lower cost and an easier introduction to programming. The Breadboard given in this package is a wiring tool used in almost all electrical experimentation capable of linking multiple different tools to accomplish almost any task. Subsequent modules after this will use the information from this packet to introduce different tools that could be used alongside the Raspberry Pi and the breadboard culminating in a final project of your choosing with additional parameters that are available in the last module. Each one will also provide a task to be completed to test your understanding of the tools and how to operate them.

Each module serves as a learning unit to instruct and challenge you to learn more about utilizing the Raspberry Pi and the breadboard to program and command a specific tool to test a specific concept and simply experiment with the provided motors and other tools. These modules will cover one distinct tool for experimentation and a basic wiring schematic to allow the Raspberry Pi to communicate with the tools properly. Each module will also give you tips on what you can do to combine 2 different tools to mark or measure a different aspect of the natural world.

Each tool within this kit varies from one to the other in terms of specs, purpose, and usages. The modules will cover various motors such as a Stepper motor or Servo motor to various sensors like a thermal sensor. Within each document will also be example code you will use for your controller to properly interact with the Breadboard and the tool used for experimentation. Each step will also have example images for reference when assembling the Raspberry Pi and the given tool. As stated before, this first module is the base on which all other modules will work off of. After completing all of the modules in this package you will be tasked with a final project that assembles all that you have learned from the previous modules to accomplish one final goal. After learning all the tools and sensors one can now apply them to potentially anything that could be measured or recorded.

Module 01:

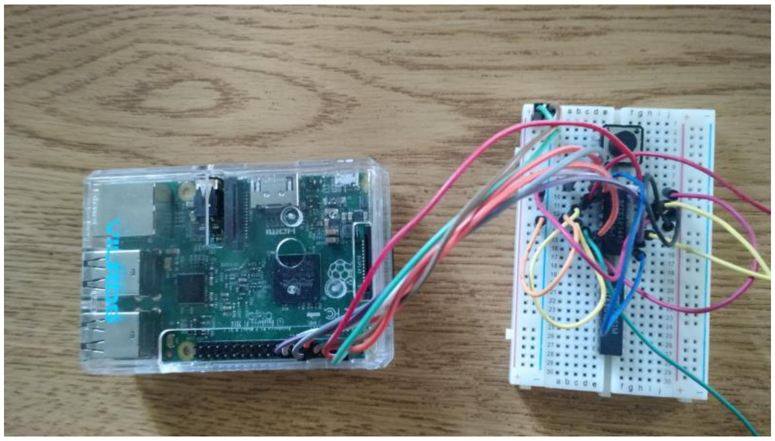
Introduction to Microcontrollers

To Measure and/or Control Physical States or Components

Module 01: Table of Contents

Objectives for Controller Module 01:

This introductory module instructs you on the Raspberry Pi and the Breadboard; showing how to setup and use the controller, the platform and how to go about connecting them together to complete a simple circuit similar to the one shown below.



Raspberry Pi connected to a Breadboard with H-bridge installed

After the introduction there will be informational passages instructing the reader on how each individual component works. With the Raspberry Pi there will be an in-depth look at the pins in the Raspberry Pi and how they interact with the breadboard and components installed within it.

Goals for the Student:

By the end of this module the Student will be able to:

- Install from scratch the Raspberry Pi OS system
- Connect the Raspberry Pi to the breadboard and the LED
- Program the LED to turn off after a set amount of time
- To be able to receive information from the Raspberry Pi

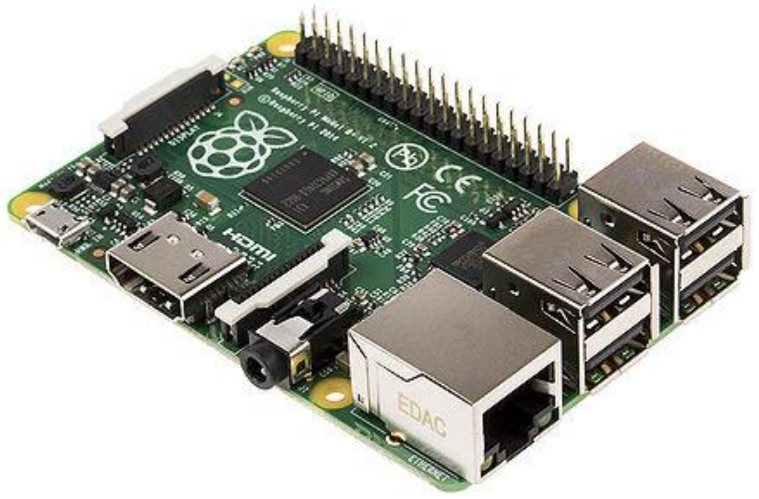
Measurable Outcomes:

By the end of the module the Student will hopefully be able to:

- Turn off and on the LED
- See when the LED would turn off and on
- Be able to install and update a micro controller OS system
- Use micro controller to write programs and routines suitable for sensing to controlling
- To be able to configure/wire an LED circuit to control illumination, both intensity and duration
- To be able to sense or detect a change in the circuit to turn on or off the LED

Introduction:

The Microcontroller:



[4]

Fig.1 Raspberry Pi model 2

The PI:

In 2006 an idea came about to give students, especially kids a new way of interacting with computers. Early concepts of the Raspberry Pi (Fig.1) were based off of the micro controller ATMega644. The microcontroller is a computer processor with a reduced size designed for embedded applications in complex devices such as the remote or appliances. Seeing how this technology can replicate the functions of a computer and be morphed into a smaller size, the foundation trustee Eben Upton[5] assembled a group of teachers, academics and computer programmers to develop a computer that will get kids interested in computer sciences.

After the goal of creating a smaller computer came into being, inspiration and prototyping started for the new invention. The physical frame of the un-named Raspberry Pi drew inspiration from ACORN's computers' BBC Micro 1981. The BBC Micro made a name

[4]Raspberry Pi

[5] Moorhead, Joanna

for itself for being a pretty reliable computer, getting bought up by the educational market. Almost 80% of the UK school system bought the Micro for use in computer classes. This invention still had a lot to be desired for since the price of the computer was £235 for Model A and £335 for Model B. This led the Founders of the Raspberry Pi to try to make the hardware smaller and more affordable for the same educational purposes than the Micro.

On the 29th of February, The Raspberry Pi foundation came out with their first, model a, deck of cards sized microcomputer. Right out of the gate the Foundation had thousands of orders from various groups of buyers, from educational customers to individual people who wanted to try it out. In subsequent years the Raspberry Pi Foundation got recognition and achievements for its first Model A device. Multiple technology writers like Glyn Moody[6] and Stephen Pritchard[7] described the Raspberry Pi as the "BBC micro 2.0" and praised it for its innovations and potential to reach and inspire the minds of children. This small computer had achieved its goal of being a cheaper introduction to the world of computer programming.

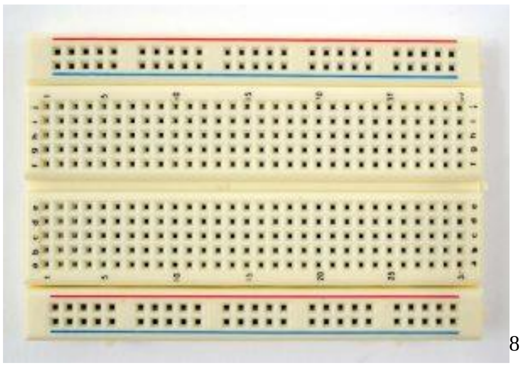

[8]

Fig.2 the Breadboard

[6] Glyn Moody,
[7] Pritchard, Stephen

The Breadboard:

A breadboard (Fig.2) is a base containing metal clips on the inside used to construct electrical circuits. This oddly named tool came about when people used to use cutting boards as a platform to hold their electrical circuits. Before the "breadboard" was invented creating a circuit board required a more resourceful mindset; using a piece of wooden board and some thumb tacks, a person had to find ways to create their own breadboard. Other tactics include using a paper schematic and lining them up on the wooden board as a guide. These were not only amateur to put together but not very conducive to experimentation. The wires were attached to some thumbnail or conductive pin and soldered into the wooden board. Since the wires and pins could not be easily removed, this made mistakes difficult to fix.

The new breadboards we have today (shown in ingredients list) was designed by Ronald J Portugal of EI Instruments Inc. in 1971 to replace the cutting boards of the past. The breadboard is solderless, meaning that you can make a circuit without having to solder the wires into place. These breadboard have different Bus strips that connect each row with an electric railway under the plastic cover. With holes for wires. This setup allows for multiple wires to be put into and taken out of the breadboard at any time. This also works in reducing the amount of wiring needed for a complete circuit and making it easier to make a circuit with a lot of miniature components such as buttons and H-bridges to get the same result as larger scaled circuits. These miniatures circuits are easy to assemble and disassemble, making the breadboard one of the most versatile electrical items in use today.

[9]

Fig.3 NTE1749 Integrated Circuit

The Integrated Circuit:

The Integrated Circuit (Fig.3) is simply an advanced electrical circuit. Despite the small size the integrated circuit holds multiple components that would usually going into a computer circuit, making it adaptable for different applications within an electrical appliance. The components have different behavior and different jobs that can be filled. The transistor acts like a switch for electricity, it can turn it on and off, or it can amplify currents. Transistors are used in computers to store information and in stereo amps to make the sound signal stronger. The resistor limits the flow of current going into or out of the circuit depending on its placement; these are used inside of TV’s and radios to control the sound volume. The capacitor collects energy within the circuit and releases it on command for usually simple purposes such as providing power for the flash of a camera. The diode stops electricity under some conditions, if those conditions change then the diode will let the energy flow. This is especially useful for preventing sparking within the circuit or control the amount of current going through a specific area of the circuit. These individual components come together to create a versatile tool to make almost any electrical system from burglar alarms to computer processors.

[9] Nte electronics

Fig.4 Close up Transistor

The transistor (Fig.4) first came about in response to an engineering problem. For the development of computers the transistor is the most important component to make modern computers run. Before the Transistor, the Vacuum Tube was the go to component for controlling electricity. The Vacuum Tube operates exactly similar to the transistor but this piece of technology is much larger than the transistor, and it is much more prone to burning out and slowing down the flow of electricity. This problem was solved and successfully demonstrated in the December 23, 1947 when three individuals credited with the invention of the transistor were William Shockley, John Bardeen and Walter Brattain. William Shockley was the man with the theory behind it, but could not build the first transistor without the help of John Bardeen and Walter Brattain[10]. These men went to work to create the first "point-contact" resistor, but the evolution did not stop here as the men came together to make a new "bipolar" transistor that would soon become part of a larger electronic device: the integrated circuit.

Integrated Circuits are microchips that have grown quite a bit from their origins. From simple transistors they include more components that increased their usefulness in daily technology. These chips can work as amplifiers and oscillators or a timer on various appliances.

[10] Brinkman

Different chips now come in two different types now based on the overall circuit: Analog or Digital. This Experiment will use an analog circuit for its simplicity and versatility to use for controlling the motors

Python:

No controller can function without a way to “communicate” with it, Python is the most common way to communicate with the Raspberry Pi. Python is an open-source programming language that is resident in with the Raspberry Pi Operating System that allows for the command of the ports on the Raspberry Pi and any subsequent tools that are connected to the Pi. Python boasts simple debugging software and easy readability for new users, alongside being built to handle multiple modules and packages for versatility. [11]

[11] python.org

Installing the Raspberry Pi software:

The first step is to get access to a computer with an SD card reader. The main site for the Raspberry Pi (raspberrypi.org) has a step by step instruction for installing the Raspberry Pi OS onto your micros SD then you may plug the micros SD into the Raspberry Pi for use. For efficiency the step by step instructions for putting together the Pi follows as such:

1. Begin by placing your SD card into the SD card slot on the Raspberry Pi (Fig 5). It will only fit one way.
2. Next, plug your keyboard and mouse (Fig 11) into the USB ports on the Raspberry Pi.
3. Make sure that your monitor or TV is turned on, and that you have selected the right input (e.g. HDMI 1, DVI, etc.).
4. Connect your HDMI cable (Fig 10) from your Raspberry Pi to your monitor or TV.
5. If you intend to connect your Raspberry Pi to the internet, plug an Ethernet cable into the Ethernet port, or connect a Wi-Fi dongle to one of the USB ports (unless you have a Raspberry Pi 3).
6. When you're happy that you have plugged all the cables and SD card in correctly, connect the micro USB power supply. This action will turn on and boot your Raspberry Pi.
7. The Raspberry Pi has an Ethernet port, alongside the USB ports. If your Raspberry Pi is situated close to a router, access point, or switch, you can connect to a network using an Ethernet cable.

8. Once you've plugged the Ethernet cable into the Raspberry Pi and the other end into an access point, your Raspberry Pi will automatically connect to the network.

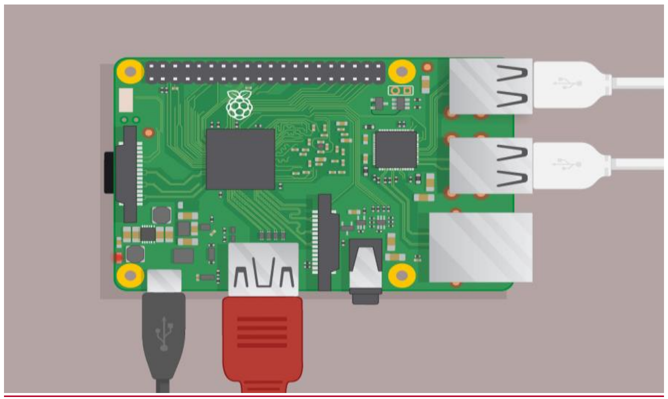


Fig 5 Raspberry Pi top down view

Fig 6 Ethernet Cable

If you followed these instructions correctly the monitor that you are using should show the insignia of the Raspberry Pi, a raspberry, to show that it is working. The first few of lines that should appear on the screen will ask for a username and a password. The username is pi and the password is raspberry for entry into the microcomputer. From here you can operate the Raspberry Pi like it was any other computer, with access to the internet, along with the python program.

Experiment: Lighting up the PI!

Materials:

Access to a computer with a SD card reader

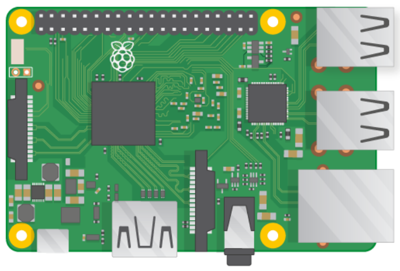


Fig 7 Raspberry Pi model 2

Fig 8. 8 GB micro SD

Fig 9 Ethernet cable

Fig 10 Monitor with hdmi input

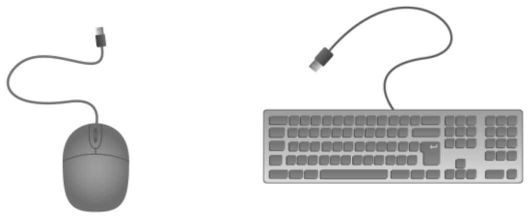
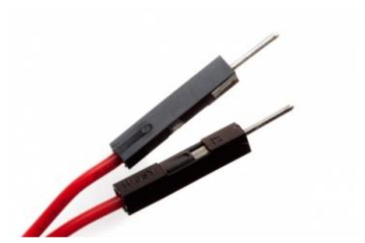


Fig 11 Wired/wireless USB mouse and keyboard

[12]Raspberry Pi
[13] Memory card readers
[14] Ethernet cable
[15] ebid

[16]

[17]



[18]

Fig 12 Female/male wires

Fig 13 Male/male wires

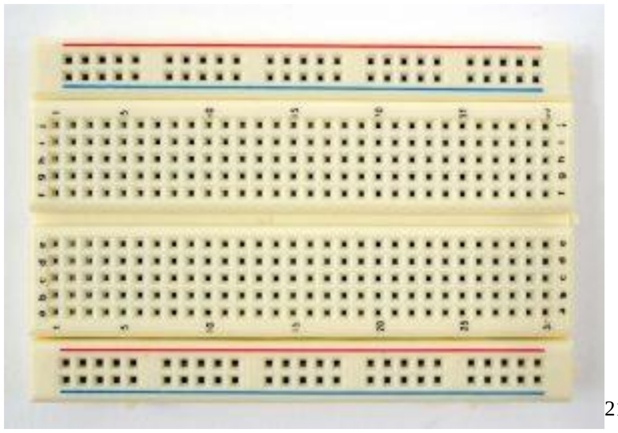
[19]

[20]

Fig 14 Power supply

Fig 15 NTE1749 Integrated Circuit

[16] Computer Cable Store
[17] Sparkfun Electronics
[18] Male to Female Jumper Wire
[19] Raspberry Pi
[20] Nte electronics

[21]

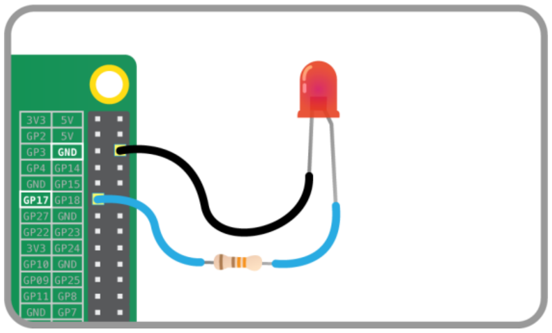


Fig 16 Bread board

Fig 17 LED light

[21] Electronic Circuit Theory and Breadboarding

Connecting Breadboard:

The Raspberry Pi has multiple ports to use in conjunction with the Breadboard to program almost anything. The Raspberry Pi has its ports set up in the configuration shown below, with the wires one would be able to make a rudimentary electrical circuit that can connect to motors or other devices to make it possible for the Raspberry Pi to communicate with it.

For the purpose of this experiment you are going to be looking at various ports like 3V, Ground, and GPI0. On both sides of the ports there are 2 ports that supply power through them directly from the Raspberry Pi, the 3 volts (port 1 and 17) and 5 volts (ports 2 and 4) all labeled in Fig 18. These ports will go directly into the breadboard to power the circuits. The GPIO (general purpose input output) ports on both sides are the direct links that the programmer uses to control different outputs from the Raspberry Pi. These ports are linked into the breadboard to control the different motors that you will be using later.

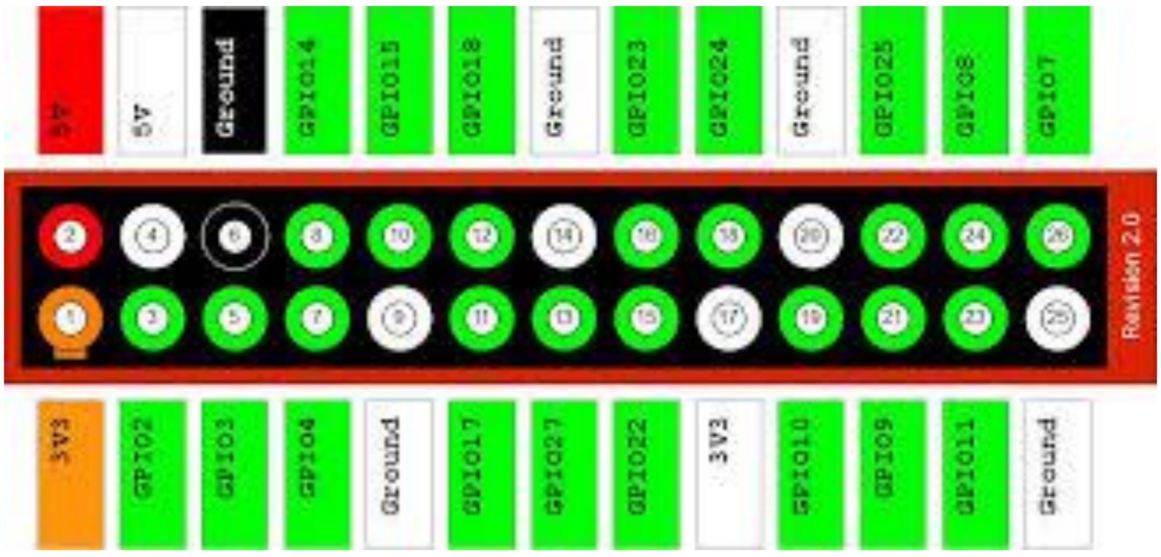


Fig 18 Pins of a Raspberry Pi

The breadboard as told before is a base for the electrical circuit to be built off of. The inside of a breadboard has metal plates that connect across rows to connect the wires placed in the holds shown in the picture below (Fig 19). By bridging the metal plates you can create different series and parallel circuits to create more complex circuits. For the experiment there will be step by step instructions for linking the Raspberry Pi to the breadboard. There is a divide in the breadboards to allow for up to five components on each side to optimize the amount of components you can install on the breadboard.

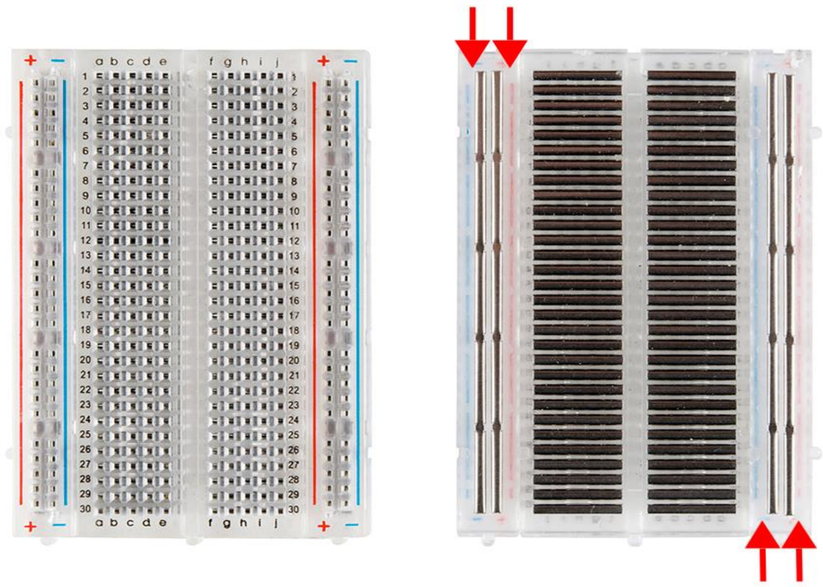


Fig 19 Breadboard with the pin separated top and the inside rails view

The first thing you should know how to do is connect the power to the breadboard. The breadboard takes in power from the Raspberry Pi through the power ports 1 or 2 and connect directly to one of positive power tracks on the left or right of the breadboard. These power rails will direct the energy down the line so that electricity can be diverted with wiring to power different components on the breadboard. This setup works similar to all other electronics, with a positive electronic input and a negative electronic input on both sides that corresponds with one or two power sources. The power ports on the Raspberry will connect to the positive power rails via the male/female wires to supply the power. The negative output or ground will connect back to the

The ground ports in the Raspberry Pi. This connection is going to be made with the female/male wire but the ground port could be any of the ones already on the Raspberry Pi. With this the Raspberry Pi can now power the breadboard and the components. For the experiment in this first module the student will be running an led light on the breadboard and lighting it up with the Raspberry Pi

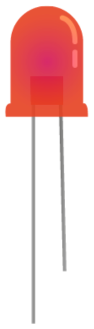


Fig 20 LED

The next thing to wire up is the actual LED (Fig 20) which is handled in 2 parts. The LED or Light Emitting Diode is a diode with a head that glows when electricity runs through it. The longer leg of the diode is the positive side of the diode (called the anode) that is always connected to the positive side of the supply, while the shorter side (called the cathode) is connected to the negative side of the power supply known as the ground. Now for the LED to work it needs a resistor to control the supply of power throughout the circuit

When using LED's and a microcontroller you must always resistors in the circuit. The Raspberry Pi can only supply the circuit 60 milliamps which is not enough for the LED. The LED will also try to draw more power out of the Raspberry Pi potentially causing the Pi to short out and potentially damage the microcontroller. The resistor stops the LED from drawing too much power from the microcontroller; specifically it limits the amount of current that is allowed to flow. The measure of this resistance and the classification of this component is called Ohms (**Ω**). This is marked on the resistor by the colored bands on the sides of the resistor body. For this experiment you will be using a 330 ohm resistor marked by:

- 4 colored bands that go Orange, Orange, Brown, then Gold
- 5 colored Bands that go Orange, Orange, Black, Black, Brown

The way that you plug in the resistor does not matter since it will reduce the flow no matter what.

Wiring the Raspberry Pi:

Steps:

1. Attach port 1 (3V) to a positive rail on the breadboard using a female/male jumper wire
2. Insert one side of a 330Ω resistor directly into the positive rail (in a parallel hole to the jumper cable just placed)and one side into another rail on the breadboard
3. Insert the anode (longer pin) of the LED on the same rail as the resistor and the cathode (shorter pin) on a separate rail
4. Connect port 9 of the Raspberry Pi to the same rail as the negative pin on the LED with a male/female jumper wire, completing the circuit

Testing the Connection:

Following the instructions above, the Raspberry Pi should be supplying power to the breadboard and the power should run through the circuit. This should supply power to the led causing it to light up. If the LED is not lit it simply means that the LED is plugged in reverse and should switch the pins of the LED so that they are plugged in the opposite way than it was before. If the LED is lit then you have successfully created a circuit

Experiment Variables:

Within the appendix you will find all of the codes for each component in the kit to use as you go through each module. For this first module please read through **Appendix Section 1** or the introductory code lesson and the sample code along with it. After inputting the code for the Raspberry Pi code you should have rudimentary control of the LED. The function will run but you will experiment with the “time.sleep()” function. This will allow you to change how long the LED stays on. Now the experiment will have you changing the resistor on the breadboard and the time in the code.

Try switching out the 330 Ohm resistor for a larger or smaller ohm resistor, you can see which resistors are which with the chart provided below.

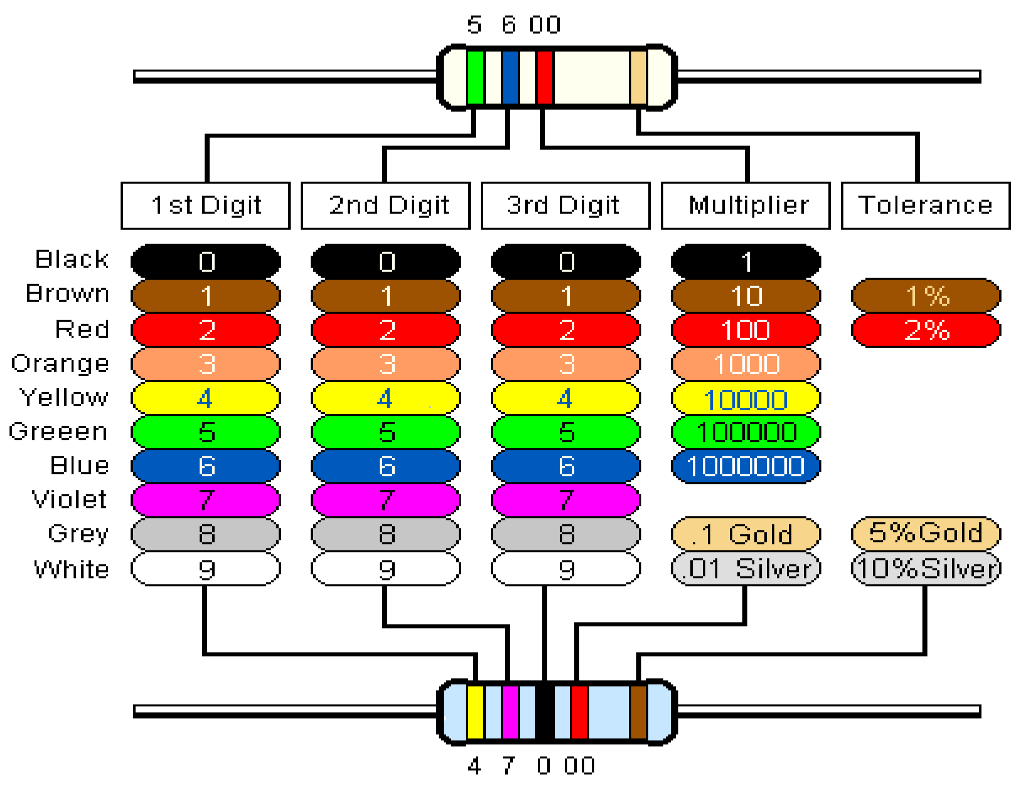


Fig 21 Resistor Chart

What happens when you switch to a larger resistor? How about a smaller resistor?

Now to change how long the LED stays on. The line of code that affects how long it stays on is the “time.sleep()” function. This function keeps track of how much time passes before it goes onto the next line of code. The function keeps track of time in seconds so putting in a whole number will be the number of seconds that the code pauses before continuing the code. Adjust the code for 5 seconds and compare the time with a stopwatch or a counting app. Adjust the code for 30 seconds and compare with a stopwatch to see if the code does as it says.

Assignments:

Write down below the scientific observations you made about the Raspberry Pi. What were some difficulties that you encountered and where was it most difficult for you?

Bibliography:

1. Johnson, G. W., & Jennings, R. (2006). LabVIEW graphical programming.

2. Brinkman, W. F., Haggan, D. E., & Troutman, W. W. (1997). A history of the invention of the transistor and where it will lead us. *IEEE Journal of Solid-State Circuits*, *32*(12), 1858-1865.

3. 1 Meter (3.28 FT) High Speed HDMI Cable with Ethernet." *Computer Cable Store*. N.p., n.d. Web. 02 Dec. 2016.

4. "Raspberry Pi." *Raspberry Pi - Raspberry Pi Hardware Guide Requirements | Raspberry Pi Learning Resources*. N.p., n.d. Web. 02 Dec. 2016.

5. #82931, Member. "Jumper Wires Premium 12" M/F Pack of 10." *PRT-09385 - SparkFun Electronics*. N.p., n.d. Web. 02 Dec. 2016.

6. SanDisk®. "Memory to Capture the Moment." *Memory Cards & Readers*. N.p., n.d. Web. 02 Dec. 2016.

7. "Ethernet Cable." *Related Keywords & Suggestions for Ethernet Cable*. N.p., n.d. Web. 02 Dec. 2016.

8. @creator_username. "Male to Female Jumper Wire." *Http://www.cytron.com.my/*. N.p., n.d. Web. 02 Dec. 2016.

9. "Electronic Circuit Theory and Breadboarding." *KachailF.weebly.com*. N.p., n.d. Web. 02 Dec. 2016.

10. "Ic-push-pull Four Channel Driver 16-pin DIP By NTE Electronics (NTE1749-1) MCM Part #NTE1749-1." *NTE Electronics Ic-push-pull Four Channel Driver 16-pin DIP | NTE1749-1 (NTE17491) | NTE Electronics*. N.p., n.d. Web. 05 Dec. 2016.

11. NTE. "NTE1749 Integrated Circuit Push–Pull Four Channel Driver." *NTE1749*. NTE Electronics Inc., n.d. Web. 16 Nov. 2016.

12. Cellan-Jones, Rory (5 May 2011). "A £15 computer to inspire young programmers". BBC News.

13. Moorhead, Joanna (9 January 2012). "Raspberry Pi device will 'reboot computing in schools'". *The Guardian*. London. Retrieved 20 January 2012.

14. National instruments. (n.d.). NI USB-6229 BNC 16-Bit, 250 kS/s M Series, Integrated BNC, External Power. Retrieved October 22, 2017, from http://sine.ni.com/nips/cds/view/p/lang/en/nid/203866

Module 02:

Motor Control Experimentation

Working with Different Types of Motors

Module 02: Table of Contents

Objectives for the Motor Module 02:

This module seeks to give an introduction to the design and functionality of multiple different types of motors. Currently in today's market, technological ingenuity keeps birthing a plethora of different tools that are used in our day to day life, and motors are some of the most basic and most important tools used. Different motors have been made to accompany different purposes for industrial uses. These motors are defined into 2 sections: AC motors and DC motors. This module will go over for the ins and outs of these sections for the student to learn about the various classifications of each one.

The components used in this module are the Raspberry Pi, breadboard, a Stepper motor, and an H-Bridge motor driver plus some previously used switches and wiring. This module will go through the step by step installation of the stepper motor to make it controllable by the Raspberry Pi. This module will also help advance your programming skills, specifically using Python and how to properly read it.

Fig 22 Stepper motor

Goals for the Student:

By the end of this module the Student will be able to:

- Connect the Raspberry Pi to the breadboard
- Connect a motor to the breadboard
- Program the motor to turn
- Turn the motor shaft to designated angles
- Change the direction of the motor multiple times in the same program
- Have the motor turn one direction, stop for a certain amount of time and turn again

Measurable Outcomes:

By the end of the module the Student will be able to:

- Get the motor to rotate in 2 different directions
- Have the motor rotate one way, stop for some period of time and rotate another way
- Connect the motors to a tread system and test out the accuracy of the motors

Introduction:

What is a motor?

The purpose of this module in this program is to give a proper introduction to motors. The motor has been a life changing event ever since the early invention of the first electric motor in 1834 by Thomas Davenport. This early invention did not output enough work for industrial usages but it showed promise by running smaller items like toy trolleys and toy trains on a small rail. As people saw this strange invention, engineers pushed to make this small machine viable for industrial usage. This challenge was first completed by the German engineer Moritz Jacobi in May of 1834 with the creation of the rotating electrical motor.

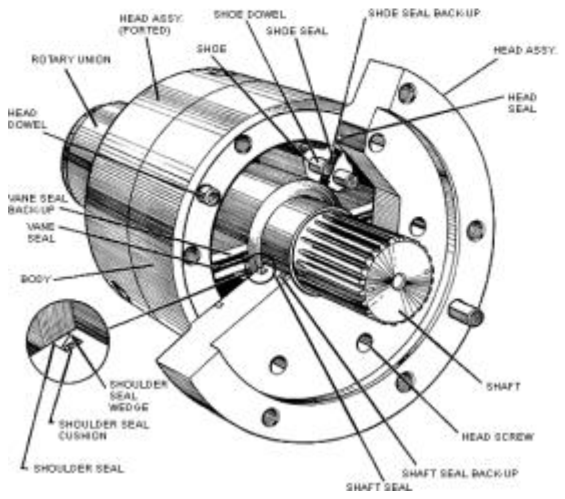


Figure 23 Basic Motor disassembly

This motor was further bested by Jacobi himself in 1838, this new motor had enough output to move a boat filled with 14 people across a river. This feat and more started the push for electric motors to be used for practical and industrial usages.

A motor, by definition, is a machine powered by electricity or internal combustion that supplies mechanical power for a vehicle or other device with moving parts. A motor in its basic form is a tool to provide work and movement to a system to accomplish a goal. The mechanical energy provided by most motors comes off as a rotational movement in one given direction at a speed based off of the amount of energy being supplied. The motor has evolved to suit many different uses, as such many different types of motors have been created to suit various needs, and this module will go into how some of these various motors work. Motors can be classified under two modes of power: AC and DC motors. These two classifications and their sub-classifications are based on the motors function or operation.

AC motors:

An AC motor runs on an Alternating Current to create motion. The AC motor converts electricity into mechanical energy through the use of magnetic fields. Magnetic fields are created inside of the motor by using two parts called the rotor and the stator. The stator holds fixed magnets that surround the motor shaft. Wires coil around the fixed magnets on the inside so that the electrical current runs directly to them. When the magnets are supplied with the alternating current, they create a magnetic field inside the motor. The rotor has a ring of magnets, or in some cases just 2 magnets, shown below, attached directly to the motor shaft that move when the magnetic field is created.[22]

[22] Woodford, C.

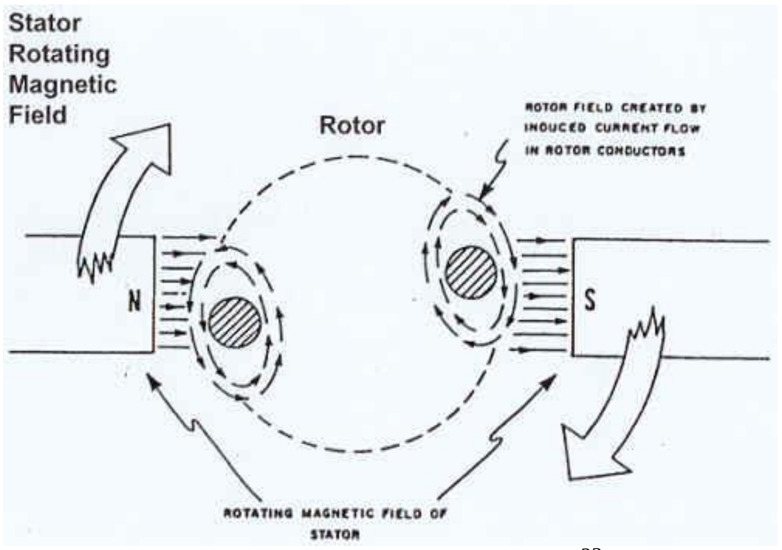

Stator Rotating Magnetic Field
Rotor
ROTOR FIELD CREATED BY INDUCED CURRENT FLOW IN ROTOR CONDUCTORS
N
S
ROTATING MAGNETIC FIELD OF STATOR

Fig 24 Basic diagram of an AC motor[23]

When the motor receives energy the rotor moves in response to the magnetic field causing the motor shaft to spin in a given direction. AC motors vary in their internal mechanisms so there are 2 different types of AC motors used in the world: Synchronous and Asynchronous motors. Synchronous AC motors works directly off the AC current. The motor shaft rotates at the same frequency as the current that is being supplied to the motor. This frequency within the motor is created by making a rotating magnetic field. Rotation of the magnetic field is accomplished with a DC power supply on the rotor to make the rotor into its own magnet to move along with the magnetic field from the stator to creating a harmonious movement. The Synchronous motor has a speed that correlates with the energy input; the faster the AC frequency the faster the motor goes.

[23] Polka, D

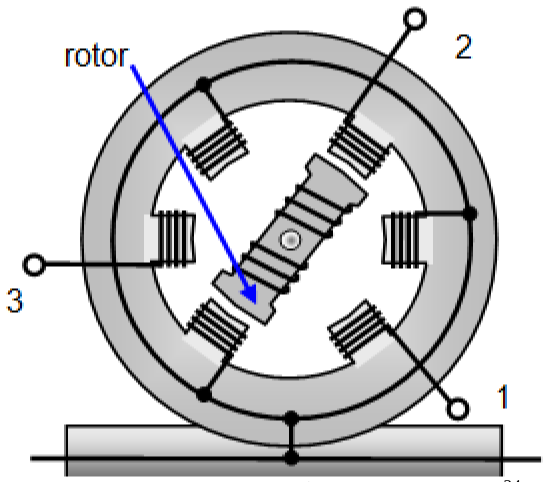


Fig 25 Basic diagram of a Sync AC motor[24]

The Non-Synchronous motor otherwise known as the Induction motor uses electromagnetic induction to power the shaft or whatever rotation device is in the motor. The way an induction motor works relies on 2 pairs of electromagnetic coils and an alternating current. Just like regular /AC motors, these motors have a stator and a rotor with the 2 pairs of coils attached to the stator and a rotor on the motor shaft. Each pair of coils is situated opposite of each other with the rotor in the middle. Since alternating current works in a flow of high and low, when one pair of coils is active, the other pair is inactive, creating specific magnetic fields that the rotor moves in response to.

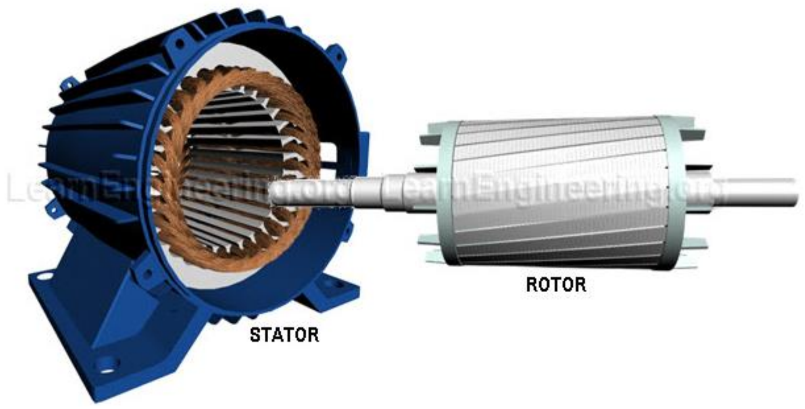


Fig 26 Basic diagram of an Induction AC motor

[24] Gibbs,K

DC motors:

The DC motor, so named since it uses Direct Current electricity to create mechanical motion. The DC motor creates mechanical movement similarly to the AC motor, by using magnetism within to create torque. This torque is created by 2 parts inside the motor: the permanent magnets and the rotating coil connected to the power supply. When an electric current passes through a coil in a magnetic field the coil will move in the direction of the magnetic force. This magnetic force and its direction define the motors capabilities

DC motors use electricity running through a wire and some metal “bristles” to create a static magnetic field. Within the wire loop is a permanent magnetic that is the polar opposite of the created magnetic field. This opposite polarity causes the magnet to spin creating mechanical energy that is utilized in whatever system it is installed in. AC motors utilize the magnetic field differently by having the magnetic field oscillate between directions, causing the magnet in the middle to move constantly to try to “catch up” with the magnetic field.[25] Each type of motor uses magnetism to produce mechanical energy but each type comes with its own strengths and weaknesses.

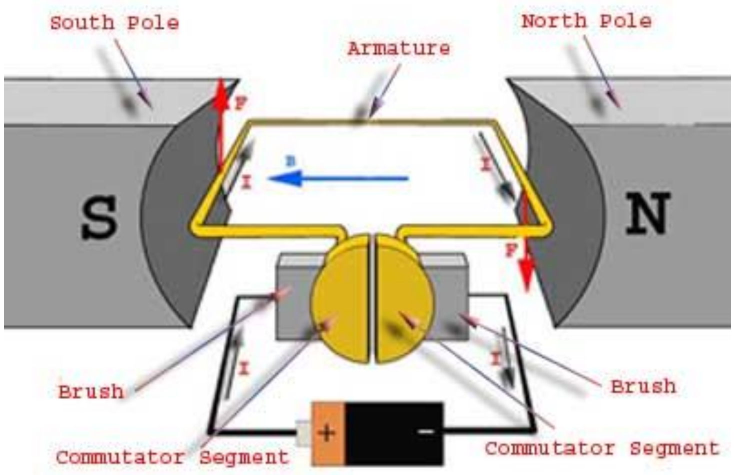


Fig 27 Basic DC Motor Diagram[26]

[25]Woodford,C
[26] electrical4u

Within the DC motor classification the Stepper Motor is one of the more dependable motors. This sensor does not have outer permanent magnets and a looping metal wire to create a magnetic field but instead has a slightly different setup. The stepper motor has multiple smaller electromagnets attached to the rotor, or the shaft that will spin, while multiple current carrying metal coils on the outside stator create specific magnetic fields. The outside metal coils are positioned to be opposite of each other so that when the current runs through it makes a positioned electrical field for the shaft to respond. The magnets on the shaft compose an inner ring that responds to the smaller magnetic fields far more precisely. With these smaller magnets the stepper is able to make “half-steps” in which you can control even further the movement of the inner ring but also control the angle in which the inner ring stops at. This ability to stop the rotation at a certain point can be called “holding torque”, allowing the rest of the body to do other things while the motor keeps its place.[27]

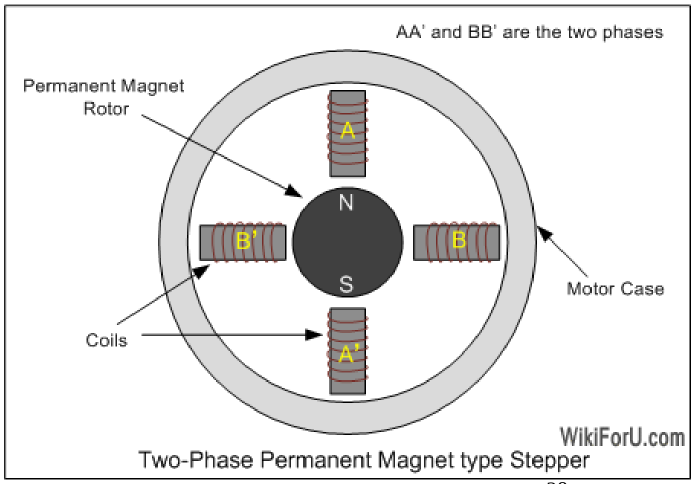

AA' and BB' are the two phases
Permanent Magnet Rotor
A
N
B'
B
S
A'
Coils
Motor Case
WikiForU.com
Two-Phase Permanent Magnet type Stepper

Fig 28 Base Diagram of Stepper Motor[28]

[27]Woodford, C
[28] wiki4u

Another reliable DC motor that will be included in the module is the servo motor. The servo motor is different from the regular dc motor because it is a compilation of the dc motor, a potentiometer, and a control circuit. This creates a new device that prioritizes position. By linking the DC motor to the potentiometer with gears the sensor will constantly measure the position, or rotation, of the motor shaft, The potentiometer relays this information to the control module which will act like a switch; the control module will supply power to the motor until the potentiometer reaches an assigned value, in which case the control stops the flow of electricity

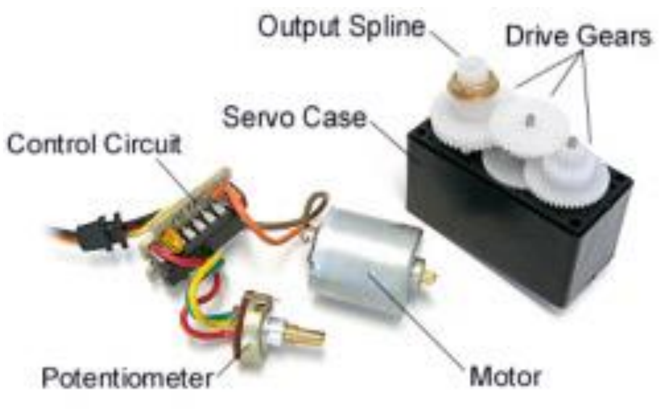


Fig 29 Diagram of Servo Motor Parts[29]

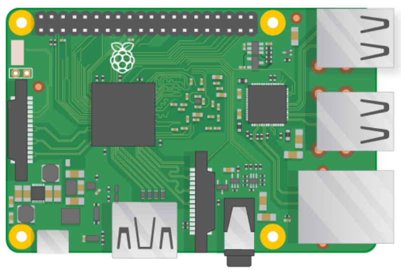
[29] Reed,F

Experiment: The Stepper Motor Escalator

For this module you will be testing out the different motors in a standard setup to compare their capabilities. The following sections will teach you how to wire up the Raspberry Pi to each motor so that the pi can control each motor. For this experiment you will be putting the given motor into the base and running a tread or thread from the motor shaft to the two other pillars to create a continuous track. This continuous track with a few markers will let you test the various properties of each motor

Materials:

Access to a computer with a SD card reader



[30]

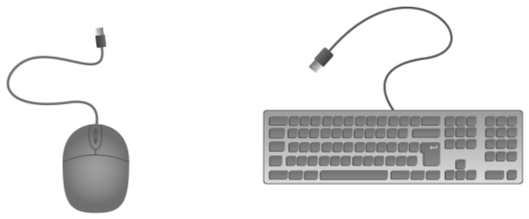
Raspberry Pi model 2

[30]Raspberry Pi

[31]

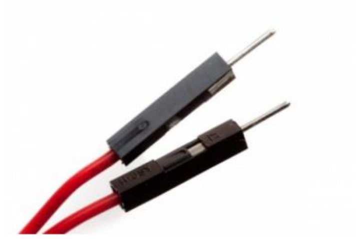
Ethernet cable

Monitor with hdmi input


[32]

Wired/wireless USB mouse and keyboard
[33]

[34]


[35]

Female/male wires

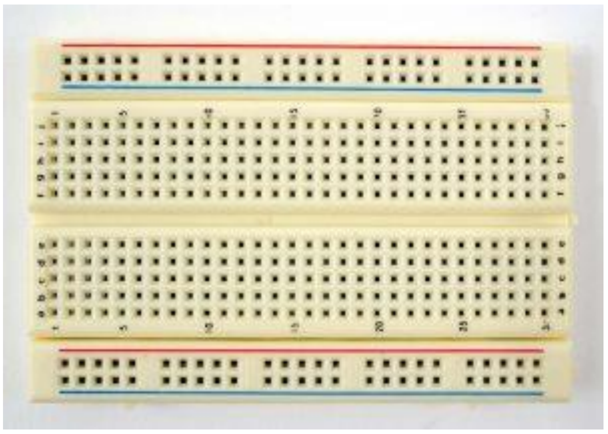
Male/male wires

[31] Ethernet cable
[32] ebid
[33] Computer Cable Store
[34] Sparkfun Electronics
[35]Male to Female Jumper Wire

36

37

Power supply NTE1749 Integrated Circuit

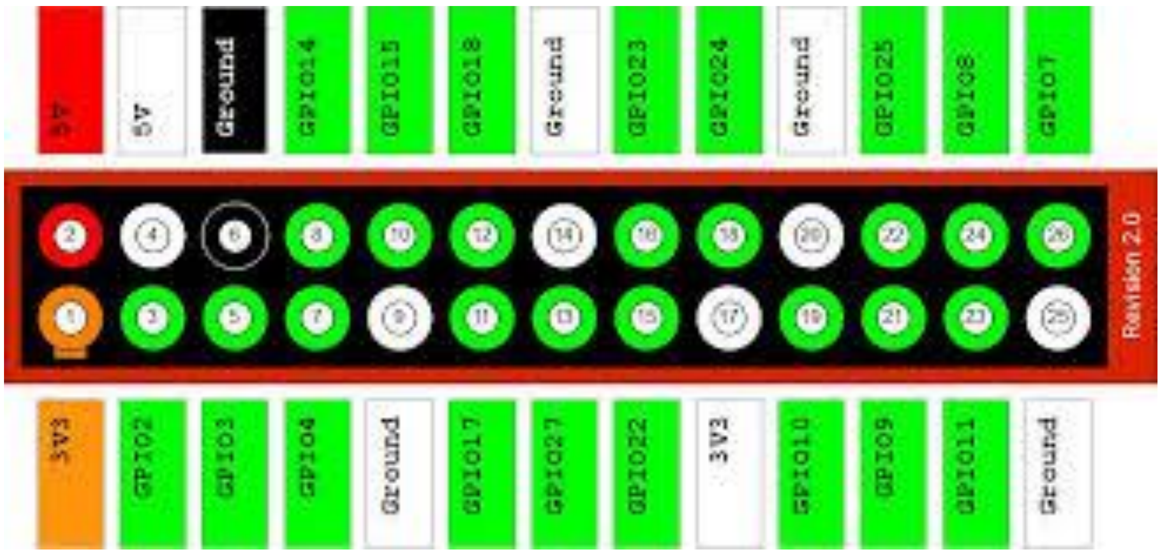


Bread board Stepper motor

[36] Raspberry Pi
[37] Nte electronics

Wiring the Raspberry Pi:

Wiring up the Raspberry Pi will work similar to the First module. Using the same layout image from before the instructions that follow will reference the image.



The Raspberry Pi will use the same ports and the same pins to supply power and ground to the breadboard so re-read the first module if power is not being sent to the breadboard. Using an LED to complete a simple circuit will tell you if the power is there. The next thing to do is to utilize the integrated circuit to connect the motor to the Raspberry Pi. The IC works similar to the Raspberry Pi with multiple pins that connect through the breadboard slots.

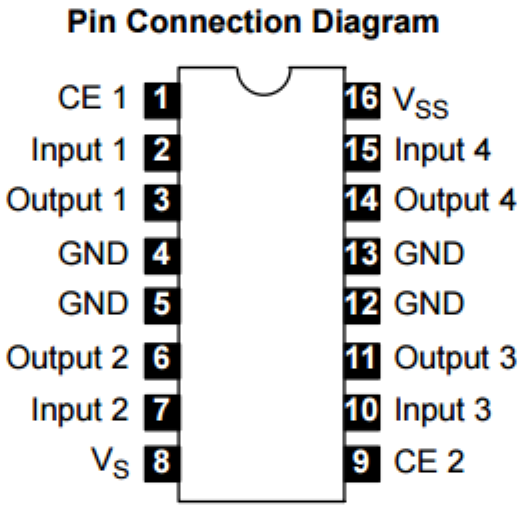


Fig 30. Integrated circuit chart

The integrated circuits are used on the divide in the breadboards to connect the two sides of the breadboards so that the components will work together. The port numbering starts from one and sixteen at the divot on the end of the microchip. The shown image dictates where the electrical inputs from the Raspberry Pi and the motors will go. When wiring up the Raspberry Pi and the motors each motor will correspond to an input and output. In the case of the experiment it will focus on one motor at a time and the Raspberry Pi. The wiring example will follow as such:

Steps:

1. Attach port 1 (3V) to a positive rail on the breadboard using a female/male jumper wire
2. Attach port 14 (Ground) of the Raspberry Pi to the negative track on the breadboard on the same side as the previous jumper wire
3. Insert the NTE 1749 microchip into the breadboard such that it is on top of the separating ravine and the pins are inserted to both sides of the breadboard
4. Now for the 4 wire motor is to be plugged in as such
    a. The black wire goes to output 3 on the microchip in one of the adjacent holes(it doesn’t matter which)

    b. The blue wire goes to output 2 on the microchip
    c. The green wire goes to output 1 on the microchip
    d. The red wire goes to output 4 on the chip

5. Now for the rest of the Raspberry Pi's port
    a. Attach port 7 on the Raspberry Pi with a female/male jumper wire and connect it to input1
    b. Attach port 9 on the Raspberry Pi with a similar wire to the negative rail on the same side as the wire from port 1
    c. Attach port 11 on the pi to the rail of pin 7 of the microchip (again anywhere is fine)
    d. Attach port 12 on the pi to the rail of pin 1 of the microchip
    e. Attach port 16 on the pi to the rail of pin 3 of the microchip
    f. Attach port 18 on the pi to the rail of pin 10 of the microchip

Platform assembly:

1. Place the rectangular platform on a flat surface to keep everything balanced and steady
2. Glue the wooden wheel to the wooden block and confirm that they stay together
3. Tape the Stepper motor onto the platform with the rotary shaft oriented parallel to the platform surface and close to one of the short edges
4. Tape the wooden block onto the platform on the opposite side of the motor with the wooden wheel facing the same direction and is parallel to the rotary shaft of the motor.
5. Attach the rubber wheel to the rotary shaft of the motor
6. Secure the rubber wheel in place with the lock gear
7. Measure and cut a length of string to loop around the rubber and wooden wheels that is taught and secure
8. Take a marker and mark a spot on the string as your measuring point
9. Place a ruler below the string to measure out distances

The Test: movement and positioning

With the platform completed and the motor connected to your Raspberry Pi, you should read **Appendix Section 2** for the given sample code to control the stepper motor. With the correct setup of the platform and the code you'd be able to get the string moving in whichever way you see fit. The given code will prompt you with the 3 questions: delay between steps, how many steps forward, how many steps backwards. The delay is how long the motor waits in between steps, this will give you time to record movements. The forward and backwards prompts are self-explanatory but you can experiment with directions on your own. Keep in mind the step ratios for the program and continue on.

Your first challenge is to align the point on the string to the lower end of the rubber wheel. The point on the string should hang on the bottom line of the string, positioned right on top of a mark on the ruler. This will be your starting position for the rest of the experiment

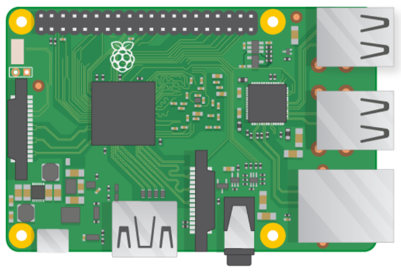
The second challenge is to find the number of steps it takes to go from one wheel to the next. Take into account the ratio and try out different numbers of steps to get to the other wheel. Using the backwards code you can reset the position of the marker by inputting the same number of steps as the forward command.

Hint: measuring how long a "step" is will go far into completing this task.

Experiment: the Model Escalator with Servo Motor:

Materials:

Access to a computer with a SD card reader



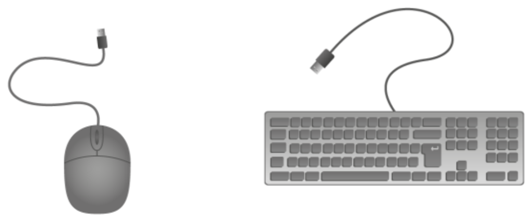
38

Raspberry Pi model 2

39

Ethernet cable

Monitor with hdmi input



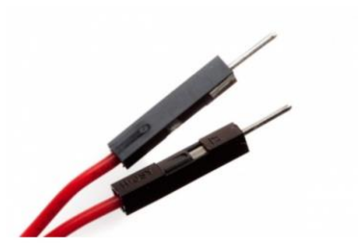
40

Wired/wireless USB mouse and keyboard

[38]Raspberry Pi
[39] Ethernet cable
[40] ebid

[41]

[42]



[43]

Female/male wires

Male/male wires

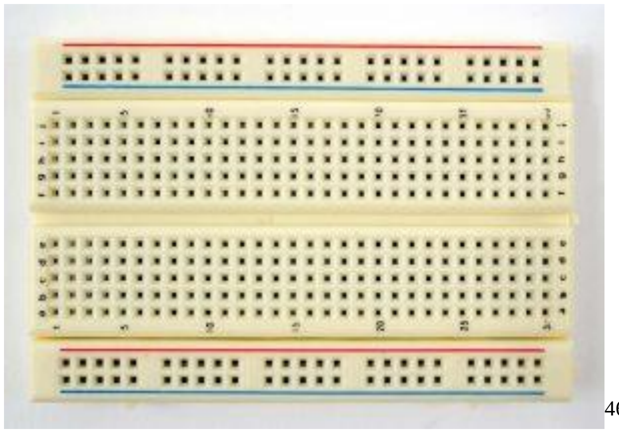
[44]

[45]

Power supply

NTE1749 Integrated Circuit

[41] Computer Cable Store
[42] Sparkfun Electronics
[43]Male to Female Jumper Wire
[44] Raspberry Pi
[45] Nte electronics


[46]

Bread board Servo motor

Wiring the pi:

Using the same schematics of the Raspberry Pi we will be wiring the Raspberry Pi to work with the servo motor. The motor itself will not work with the platform given in this kit so it will be on the side as it connects to the breadboard. For the servo motor we will not be using the integrated chip since the circuit can be completed in a simpler design.

Steps:

1. Connect pin 22, pin 6, and pin 2 of the Raspberry Pi to one side of the breadboard in separate rows so they don't cross each other
2. Connect the ground wire (the black/brown wire) of the servo motor to the same row as the wire for pin 6 of the Raspberry Pi

[46] Electronic Circuit Theory and Breadboarding

3. Connect the yellow wire on the servo motor To the same row as pin 22 on the breadboard
4. Connect the red wire on the server motor to the same row as pin 2 to get power flowing from the Raspberry Pi to the servo motor.

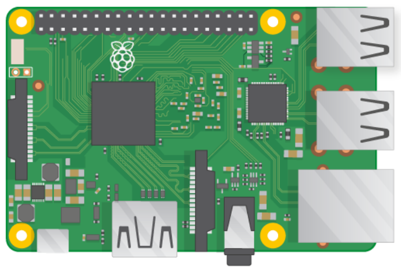
The Test part 1: movement and positioning

After inputting the code from **Appendix Section 3** try changing the angles in the code to get the servo motor to move to different locations. Marking an edge of the servo motor wheel with tape to make a bookmark. Using the extension on the platform you can put the servo motor on it and connect the wooden wheels and the treads on the assembly. Now using the code, try to measure the accuracy of the servo motor by making some measurable mark along the tread path. Experiment with how close you can get some point of the tread of your choice close to, if not directly over the mark. Record your findings and the number of attempts you had before you succeeded.

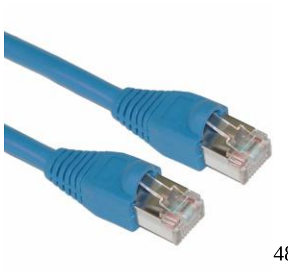
Experiment: the Model Escalator with DC Motor:

Materials:

Access to a computer with a SD card reader

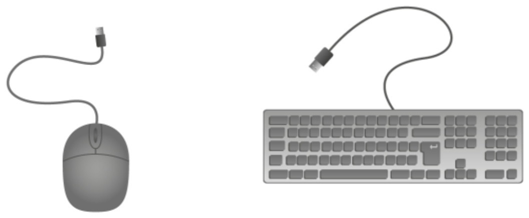

[47]

Raspberry Pi model 2


[48]

Ethernet cable

Monitor with hdmi input

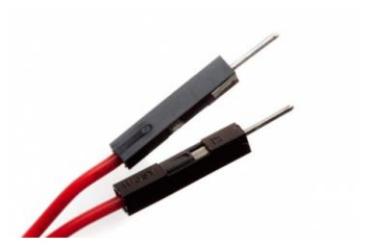

[49]

Wired/wireless USB mouse and keyboard

[47]Raspberry Pi
[48] Ethernet cable
[49] ebid

50

51


52

Female/male wires

Male/male wires

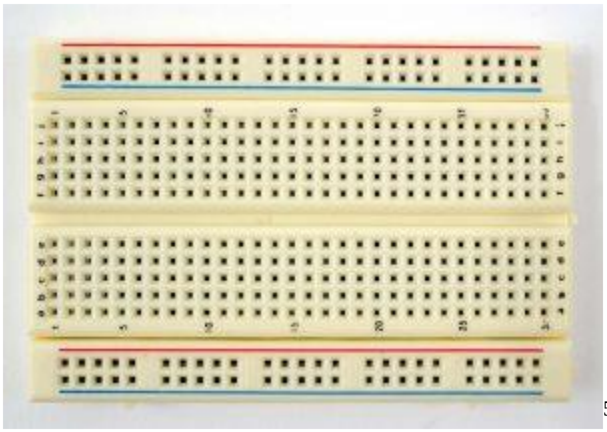
53

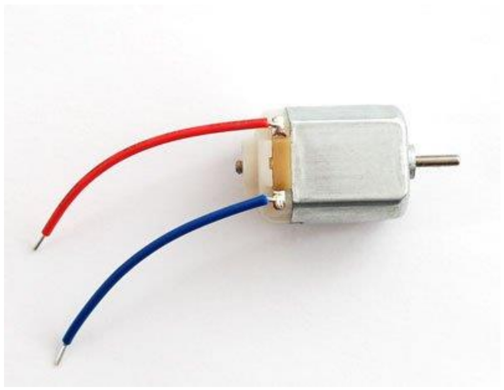
54

Power supply

NTE1749 Integrated Circuit

[50] Computer Cable Store
[51] Sparkfun Electronics
[52]Male to Female Jumper Wire
[53] Raspberry Pi
[54] Nte electronics


[55]



Bread board

DC motor

[55] Electronic Circuit Theory and Breadboarding

Wiring the pi:

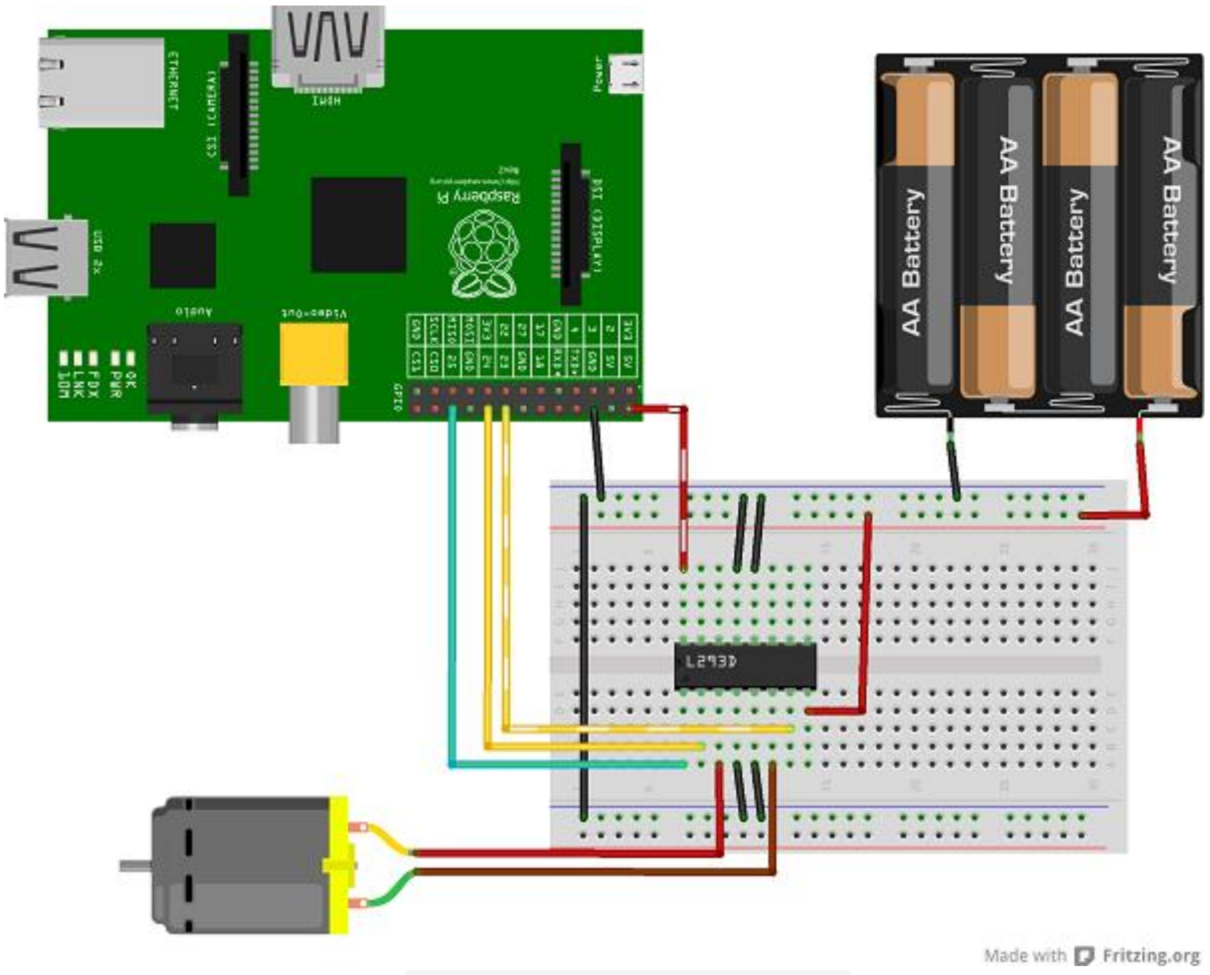


Figure 31 Completely wired motor layout

The DC motor will utilize the H-bridge chip in the setup (Figure 31). The Raspberry Pi will be supplying power to the breadboard and the dc motor as you experiment with controlling the dc motor from the microcontroller.

Steps:

1. Attach a wire from pin 2 (5V) of the Raspberry Pi to the same row as pin 16 of the h-bridge (Vss)
2. Attach a wire from pin 6 (Ground) of the Raspberry Pi to the right blue column of the breadboard

3. Attach a wire from pins 13 and 12 of the H-bridge to the right blue column.
4. Attach a wire from pin 18 (GP1024) of the Raspberry Pi to the row of pin 2 (input1) of the h-bridge
5. Attach a wire from pin 16 (GP1023) of the Raspberry Pi to the row of pin 7 (input2) of the h-bridge
6. Attach a wire from pin 22 (GP1025) of the Raspberry Pi to the row of pin 1 (CE1)
7. Attach the positive wire of the DC motor into the row of pin 3 (output1) of the H-bridge
8. Attach the negative wire of the DC motor into the row of pin 6(output2) of the H-bridge, put this wire in the same column as the positive wire of the DC motor
9. Attach a wire from pins 4 and 5 of the H-bridge (Ground) to the leftmost blue negative column.
10. Use a wire to connect both ground columns of the breadboard together to complete the circuit

Test for DC Motor: positioning and precision

Just like earlier motors you will be testing how accurate and functional the DC motor is. Copy and input the code provided in **Appendix Section 4: The DC Motor** into Python to run for the DC motor. As you probably observed there is no in-between speed for the DC motor, it is only high and low for the motors to operate at. Attach gear on the gear shaft of the DC motor to work with the tread and run on the other wheels and attempt to get a point of the tread to stop right above the same spot as the previous 2 experiments. Record your experiences and how many attempts it took to get the point of the tread to stop on top of the stop off point.

Assignment:

Each motor has different functions and different strengths and weaknesses, write down some of the Pros and cons of each motor

Stepper:

Servo:

DC motor:

Bibliography

1. Polka, D. (2001, July). AutomatedBuildings.com Article - What Is A Drive? Retrieved November 10, 2016, from http://www.automatedbuildings.com/news/jul01/art/abbd/abbd.htm

2. Woodford, C. (2016, February 12). AC induction motors | How AC motors work. Retrieved November 8, 2016, from http://www.explainthatstuff.com/induction-motors.html

3. Woodford, C. (2016, February 03). How do stepper motors work? Retrieved November 9, 2016, from http://www.explainthatstuff.com/how-stepper-motors-work.html

4. Teschler, L | Machine Design, L. (2013, January 4). What's the Difference Between Asynchronous and Synchronous Motors? Retrieved November 11, 2016, from http://machinedesign.com/motorsdrives/whats-difference-between-asynchronous-and-synchronous-motors

5. Reed, F. (n.d.). How Do Servo Motors Work. Retrieved November 11, 2016, from http://www.jameco.com/Jameco/workshop/howitworks/how-servo-motors-work.html

6. Stepper Motor | Construction, Working and Types. (2011, November 11). Retrieved November 11, 2016, from http://www.wikiforu.com/2011/11/stepper-motor-construction-working-and.html

7. Doppelbauer, Martin. "The invention of the electric motor 1800-1854." *History - The invention of the electric motor 1800-1854*. N.p., 25 Sept. 2014. Web. 25 Feb. 2017.

8. Barnett, Jason. "Controlling DC Motors Using Python With a Raspberry Pi." *Business Envato Tuts*. N.p., 11 Apr. 2014. Web. 25 Feb. 2017.

9. Gibbs, K. (2013). The a.c. motor. Retrieved November 9, 2016, from http://www.schoolphysics.co.uk/age16-19/Electricity%20and%20magnetism/Electromagnetism/text/Electric_motor_ac/index.html

10. How does an Induction Motor Work? (2013, August 6). Retrieved November 11, 2016, from http://www.learnengineering.org/2013/08/three-phase-induction-motor-working-squirrel-cage.html

11. Induction Motor | Working Principle | Types of Induction Motor. (n.d.). Retrieved November 11, 2016, from http://www.electrical4u.com/induction-motor-types-of-induction-motor/

12. Synchronous Motor Working Principle. (2013, July 16). Retrieved November 8, 2016, from http://www.electrical4u.com/synchronous-motor-working-principle/

13. CableOrganizer.com. "Three-Phase Electric Power." *Three-Phase Electric Power*. N.p., n.d. Web. 10 Feb. 2017.

14. "What is the difference between BOARD and BCM for GPIO pin numbering?" *Python - What is the difference between BOARD and BCM for GPIO pin numbering? - Raspberry Pi Stack Exchange*. N.p., n.d. Web. 25 Apr. 2017.

15. Doppelbauer, Martin. "The invention of the electric motor 1800-1854." *History - The invention of the electric motor 1800-1854*. N.p., n.d. Web. 10 Feb. 2017.

Module 3

Sensor control experimentation

Recording the world through Sensors

Table of Contents

Objectives for the Controller Module III:

This module seeks to give an introduction to the design and uses of multiple different types of distance sensors. As technology grows in uses, we as a society desire our technology to interact more with the daily physical world. Using different forms of radiation such as X-ray and radio waves, sensors have been created to produce and use radiation to record data about various distances. These sensors are defined by the way that they interact with the world, whether they use sound waves or spectral light waves. This module will cover 3 different types of sensors that use various methods to judge and interact with distance.

The components that this module will be working with are the same Raspberry Pi and breadboard of the previous modules, alongside the ultrasonic sensor, PIR sensor and PIR Motion Sensor. This module will also take you through the step by step installation of each sensor as such that the Raspberry Pi can receive information that the sensor records. This module will also contain continued insight on the format of the programming language Python and how to properly read it.

Goals for the students:

By the end of this module the Student will hopefully be able to:

- Connect the three different sensors to the Raspberry Pi
- Get the sensors to respond to outside stimuli
- Program the Raspberry Pi to receive information

Measureable Outcomes:

By the end of the module the Student will hopefully be able to:

- Get the sensors to display distance information
- Turn on a led when an object gets to a certain distance

Introduction:

What is a sensor?

The purpose of this module in this program is to give a proper introduction to multiple different types of sensors. By basic definition a sensor is a device that detects and responds to some input from the outside environment. Sensors respond to specific types of outside stimulus like light, sound or temperature. These stimuli would cause an electrical input to send a signal back to a display for the programmer to use. The first sensor, created in 1883, was an electric thermostat invented by Warren Johnson[56], which was accurate to a degree, even more than easily accessible thermostats today. This invention started a path of inventions that allowed humans another way to interact with the world around us.

With the different aspects of life to measure, many different types of sensors were invented to measure previously immeasurable properties. Sensors like smoke, temperature, humidity, and motion came through after the first sensor to suit many needs. The motion sensor first came about in the 1940s by the man Samuel Bagno[57]. Its first use was simply finding a person within a room. Since then, its potential grew as the sensor evolved to calculate the movement of the stars. The motion sensor grew more useful in various fields of life, from the land mines in World War 2 to the metal detectors used in airport security, sensors have become almost fully integrated in our daily lives

[56] Ali
[57] ebid

PIR Motion Sensor:

A PIR Motion Sensor, the shortened name of a Pyroelectric ("Passive") Infrared sensor, is one of the distance sensors that will be covered in this module. PIR sensors allow you to sense motion; it’s almost always used to detect whether an object has moved in or out of the sensors range. They are small, inexpensive, low-power, easy to use and don't wear out easily after long time use. For that reason they are commonly found in appliances and gadgets like a microwave and video camera software.

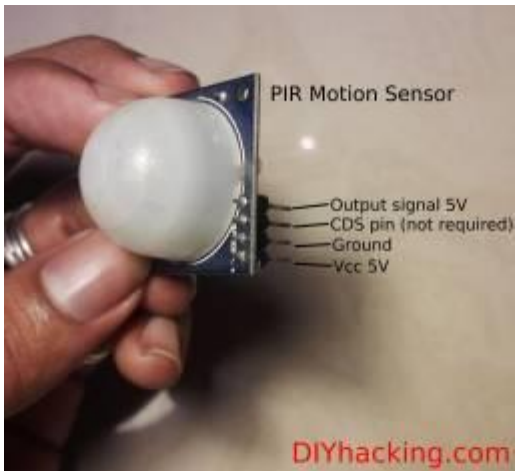


Figure 32 PIR Motion Sensor Automation

The sensor itself is made up of a Fresnel lens, an infrared detector, and supporting detection circuitry. The lens on the sensor focuses any infrared radiation/wavelengths present around it towards the infrared detector. Our bodies generate low levels of radiation as heat and as a result, this gets picked up by the motion sensor. The sensor outputs a 5V signal for a period of one minute as soon as it detects us. It offers a tentative detection range of about 6-7 m and is highly sensitive. The output from the sensor is used to trigger a transistor BC547. The transistor then switches on a 5V relay. The relay correspondingly switches your appliance ON. This module will teach the student how to program the Raspberry Pi to read this trigger and respond to it.

Ultrasound Sensor:

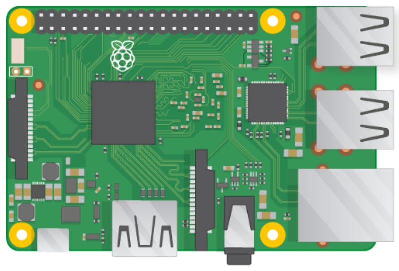
Sound consists of oscillating waves through a medium (such as air) with the pitch being determined by the closeness of those waves to each other, defined as the frequency. Only some of the sound spectrum is audible to the human ear, defined as the “Acoustic” range. Very low frequency sound below Acoustic is defined as “Infrasound”, with high frequency sounds above, called “Ultrasound”. Ultrasonic sensors are designed to sense object proximity or range using ultrasound reflection, similar to radar, to calculate the time it takes to reflect ultrasound waves between the sensor and a solid object. Ultrasound is mainly used because it’s inaudible to the human ear and is relatively accurate within short distances.

A basic ultrasonic sensor consists of one or more ultrasonic transmitters (basically speakers), a receiver, and a control circuit. The transmitters emit a high frequency ultrasonic sound, which bounce off any nearby solid objects. Some of that ultrasonic noise is reflected and detected by the receiver on the sensor. That return signal is then processed by the control circuit to calculate the time difference between the signal being transmitted and received. This time can subsequently be used, along with some mathematical calculations the distance between the sensor and the reflected object.

Experiment: Acquiring and Recording data using the Raspberry Pi!

Materials:

Access to a computer with a SD card reader

 [58]

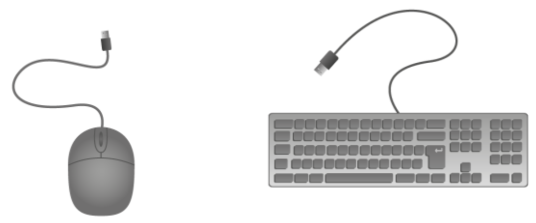
Raspberry Pi model 2

[59]

Ethernet cable

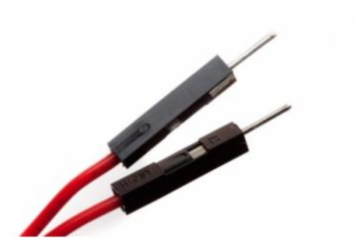
Monitor with hdmi input

[58]Raspberry Pi
[59] Ethernet cable


60

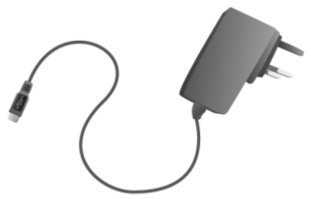
Wired/wireless USB mouse and keyboard
61

62


63

Female/male wires

Male/male wires

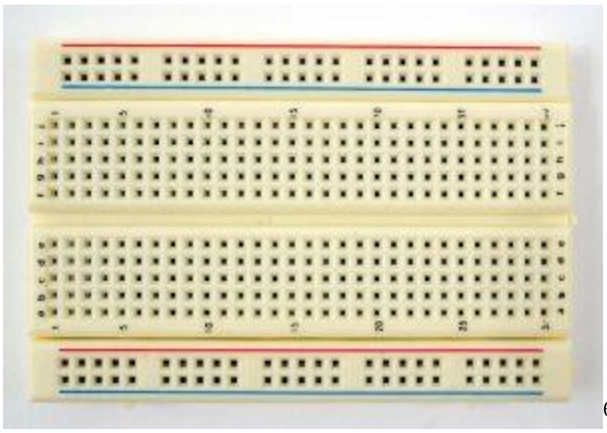

64

65

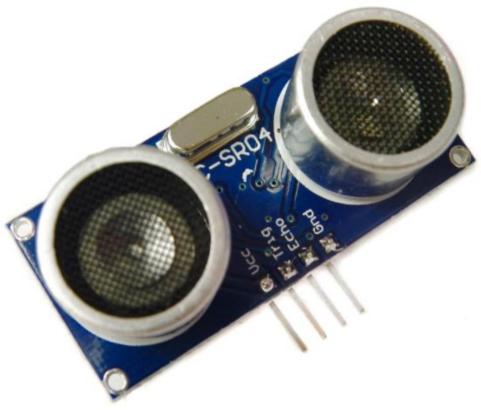
Power supply

NTE1749 Integrated Circuit

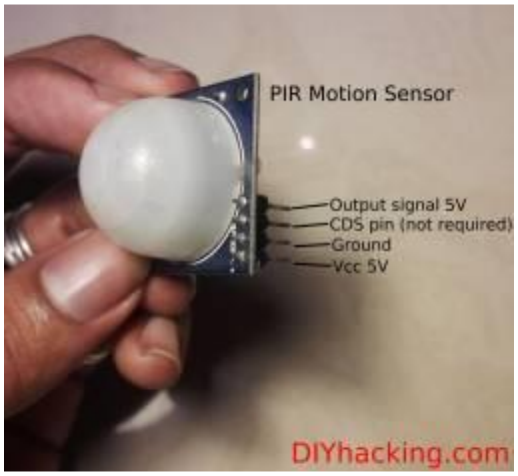
[60] ebid
[61] Computer Cable Store
[62] Sparkfun Electronics
[63]Male to Female Jumper Wire
[64] Raspberry Pi
[65] Nte electronics


[66]

Bread board



Ultrasonic Sensor



PIR Motion Sensor

[66] Electronic Circuit Theory and Breadboarding

Wiring up the Raspberry Pi for the PIR Motion Sensor:

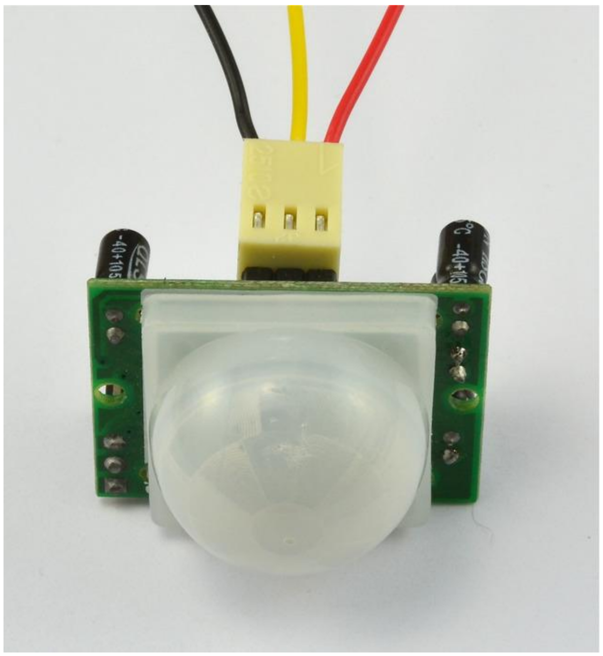


Figure 33 Pir Motion Sensor

The PIR Motion sensor (Fig 33) has 3 pins on it that connect to the breadboard. Make sure that the socket is the right way around (use the picture above) and that the red lead goes to 5V, the black to GND and the yellow to 18 on the Raspberry Pi.

Wiring Setup Steps:

1. Connect a wire from Pin 4 to a row on the breadboard
2. Connect the red lead of the PIR Sensor to the breadboard in the same row as the first wire that you put in
3. Connect a wire from pin 18 on the Raspberry Pi to a different row on the breadboard
4. Connect the yellow wire of the PIR Sensor to the same row as the wire from pin 18
5. The black wire of the sensor goes directly into the ground line of the breadboard
6. Connect the wire of pin 6 (ground) to the same line as the negative ground on the breadboard.

Testing the PIR Motion Sensor:

Copy and input the code from **Appendix Section 5** into Python for execution. This code will rely on the libraries “time” and “RPI.GPIO”. The code will print the words PIR ALARM when there is a change in distance so test if your setup works by placing something in front of the PIR sensor. If the python writer displays the text then you have succeeded.

Wiring up the Raspberry Pi for the Ultrasound Sensor:

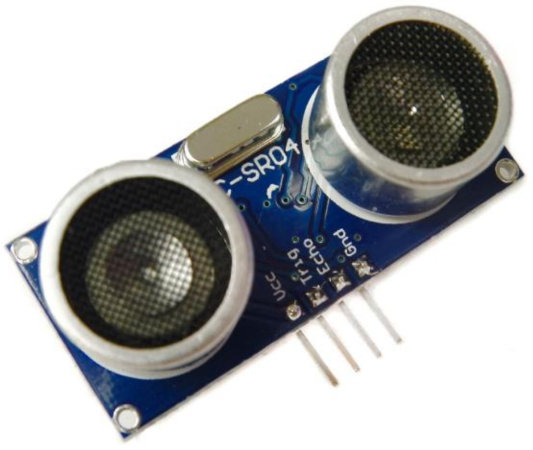


Figure 34 Ultrasound Sensor

The Ultrasound Sensor (Fig 34) has 4 pins on it: the Ground, Echo, Trigger, and the VCC pins that it uses to interact with the Raspberry Pi. The sensor draws power from the Raspberry Pi through the VCC or 5V supply to operate. Then the Raspberry Pi sends a signal through the Trigger pin to send an ultrasonic pulse through the speakers. The Echo pin is what the sensor uses to receive the sound waves bouncing back from the object.

Wiring setup steps:

1. Plug male to female jumper wires into the pins of the sensor with corresponding colors:
    a. Red wire for Vcc
    b. Blue for Trig
    c. Yellow for Echo
    d. Black for Ground
2. Plug Vcc into the positive rail of your breadboard, and plug GND into your negative rail.
3. Plug GPIO 5V [Pin 2] into the positive rail, and GPIO GND [Pin 6] into the negative rail

4. Plug TRIG into a blank rail, and plug that rail into GPIO 23 [Pin 16]. (You can plug TRIG directly into GPIO 23 if you want). I personally just like to do everything on a breadboard!
5. Plug ECHO into a blank rail, link another blank rail using R1 (1kΩ resistor)
6. . Link your R1 rail with the GND rail using R2 (2kΩ resistor). Leave a space between the two resistors.
7. Add GPIO 24 [Pin 18] to the rail with your R1 (1kΩ resistor). This GPIO pin needs to sit between R1 and R2

Testing the Ultrasonic Motion Sensor:

Copy and input the code from **Appendix Section 6** into Python for execution. This code will rely on the same libraries "time" and "RPI.GPIO" as the PIR Motion sensor. The code will print the words different words from the PIR Sensor that are changeable in the code. The most important part is that there is a reaction when a change in distance occurs.

Assignment:

When done correctly each sensor should give a visual response in the Python shell. The ultrasonic sensor and the PIR sensor have different operable distances that should become apparent with trials and testing. Display your findings in a suitable lab book

Bibliography:

1. "A Brief History of RFID." *A Brief History of RFID*. N.p., n.d. Web. 06 Apr. 2017.

2. Abdalla Ali, Member Follow. "History of sensors." *LinkedIn SlideShare*. N.p., 08 Mar. 2015. Web. 06 Apr. 2017.

3. Ada, Lady. "PIR Motion Sensor." *How PIRs Work | PIR Motion Sensor | Adafruit Learning System*. N.p., n.d. Web. 06 Apr. 2017.

4. "Introduction to the RFID." *Version française*. N.p., n.d. Web. 06 Apr. 2017.

Adafruit. "PIR Motion Sensor Tutorial." *Instructables.com*. Instructables, 12 May 2016. Web. 06 Apr. 2017.

5. Violino, Bob. "The History of RFID Technology." *The History of RFID Technology - 2005-01-16 - Page 1 - RFID Journal*. N.p., 16 Jan. 2005. Web. 06 Apr. 2017.

6. Ali, Abdalla. "History of sensors." *LinkedIn SlideShare*. N.p., 08 Mar. 2015. Web. 26 Feb. 2017.

7. "PIR Motion Sensor Automation: The Best Tutorial." *DIY Hacking*. N.p., 20 Feb. 2017. Web. 26 Feb. 2017.

8. "HC-SR04 Ultrasonic Range Sensor on the Raspberry Pi." *Cases for your Raspberry Pi*. N.p., 11 Sept. 2015. Web. 06 Apr. 2017.

9. Ada, Lady . "PIR Motion Sensor." *How PIRs Work | PIR Motion Sensor | Adafruit Learning System*. N.p., 09 Sept. 2016. Web. 06 Apr. 2017.

Module 4

Culmination Project

Putting together everything you’ve learned

Table of Contents

.

Module Objectives:

This module is the final module for this learning package. The project requires the student to implement several of the previous modules in an open-ended project. The task is to unlock a door upon approach of a person.

Goals for the students:

The goal of this module is to require the student to use the material learned from the microcontroller module coupled with some form of control, such as a motor and some form of sensing. Using these previous modules lessons, determine a way to satisfy the objectives – unlock the door when a person approaches.

Measureable Outcomes:

Design your experiment specifying the functional and design constraints.

Specify the tasks and functions that the microcontroller will perform.

Determine what control actions will be implemented.

Determine what components need to be sensed, and how they will be sensed.

Write a procedure section

Specify what needs to be measured, recorded and actions taken.

RESULTS:

Throughout the 3 terms working on this project I have successfully put together a blue print of a 4 module learning package for future students. Each module covers the assembly, programming, and application of each component that would be offered in the package. Module one fully covers what the Raspberry Pi is, how to properly wire and power the device and how to program on it. I also held the same assembly structure among the rest of the modules so the integration of the different components will be more fluid for students. The code provided in the appendix works soundly in the assignments and should direct the students in.

Conclusion:

This project has taken a lot of time, effort, and mental fortitude to complete alone, and there is much work to be done with the modules. The individual modules 1 and 2 have been tested with the code and the assembly so I have the utmost confidence in them. Module 3 was not tested, but following the format of the previous 2 modules, module 3 should follow the same logic and instruction method so that students won't become lost. Module 4 may or may not be too vague but the idea behind it supports a freedom of choice for the students to do what they want

<u>**Appendix**</u>

<u>Appendix Section 1: The Raspberry Pi</u>

With the correct setup of the breadboard, you should be able to program the Raspberry Pi to interact with the breadboard and anything you have connected on it with the program language Python. Python is a relatively simple, general programming language used wherever programming is used. There are many programming languages such as SQL (pronounced sequel), Java, C++ and more that allow for communication and programming of different components and computer systems to accomplish many great achievements. These languages are the building blocks of modern society; programs make computers and advanced systems function and grants us the ability to do so much with the smallest piece of technology.

With the variety of languages to choose from Python was chosen for its easily teachable properties. Like vocal languages learning a language takes time and practice to understand its nuances, except this relies on understanding the writing structure of the program. Python and all other languages read the program line by line and interpret that code into actions. An example of this format would be this counter code here:

```
# Store input numbers
num1 = input('Enter first number: ')
num2 = input('Enter second number: ')
# Add two numbers
sum = float(num1) + float(num2)
# Display the sum
print('The sum of {0} and {1} is {2}'.format(num1, num2, sum))
```

This code follows a certain format that python reads step by step, taking in the variables and any pre-programmed commands and then interprets the information. Python reads the lines [;based off of the indents in the lines and the order in which they happen. For example all of the code in the above example start on the complete left side of the document. Indents in the code would tell python that there are no Loops or groups of code that need to be executed. Each line has functions and variables that do different things depending on what you type in. for now the main things that will be focused on are Variables and Built in Functions.

Variables, just like in mathematics and science, are changeable elements that could be used to instruct the rest of the code with what to do. The variables in python follow this format:

Name = function(data)

The variable in this line is the “Name”. The equal sign defines the name as the result of the functions that you input. This makes it easier to link together multiple functions to get an end result. Now a Function is just the name of a procedure that the computer will run. Each function does something different; from adding up a list of numbers, to assembling a list of words beginning with the letter “h”. Each function has its use but each follows a simple structure of the function followed by “()”. The data inside along with the function will dictate the results of the computation. These modules will use multiple modules so the following list will define what each function does:

abs()	Return the absolute value of a number.
all()	Return True if all elements of the iterable are true (or if the iterable is empty).
any()	Return True if any element of the iterable is

	true. If the iterable is empty, return False.
ascii()	Return a string containing a printable representation of an object, but escape the non-ASCII characters.
bin()	Convert an integer number to a binary string.
bool()	Convert a value to a Boolean.
bytearray()	Return a new array of bytes.
bytes()	Return a new "bytes" object.
callable()	Return True if the object argument appears callable, False if not.
chr()	Return the string representing a character.
classmethod()	Return a class method for the function.
compile()	Compile the source into a code or AST object.
complex()	Create a complex number or convert a string or number to a complex number.
delattr()	Deletes the named attribute of an object.
dict()	Create a new dictionary.
dir()	Return the list of names in the current local scope.
divmod()	Return a pair of numbers consisting of quotient and remainder when using integer division.

enumerate()	Return an enumerate object.
eval()	The argument is parsed and evaluated as a Python expression.
exec()	Dynamic execution of Python code.
filter()	Construct an iterator from elements of iterable for which function returns true.
float()	Convert a string or a number to floating point.
format()	Convert a value to a "formatted" representation.
frozenset()	Return a new `frozenset` object.
getattr()	Return the value of the named attribute of an object.
globals()	Return a dictionary representing the current global symbol table.
hasattr()	Return `True` if the name is one of the object's attributes.
hash()	Return the hash value of the object.
help()	Invoke the built-in help system.
hex()	Convert an integer number to a hexadecimal string.
id()	Return the "identity" of an object.
input()	Reads a line from input, converts it to a string (stripping a trailing newline), and returns that.

int()	Convert a number or string to an integer.
isinstance()	Return `True` if the object argument is an instance.
issubclass()	Return `True` if class is a subclass.
iter()	Return an iterator object.
len()	Return the length (the number of items) of an object.
list()	Return a list.
locals()	Update and return a dictionary representing the current local symbol table.
map()	Return an iterator that applies function to every item of iterable, yielding the results.
max()	Return the largest item in an iterable.
memoryview()	Return a "memory view" object created from the given argument.
min()	Return the smallest item in an iterable.
next()	Retrieve the next item from the iterator.
object()	Return a new featureless object.
oct()	Convert an integer number to an octal string.
open()	Open file and return a corresponding file object.
ord()	Return an integer representing the Unicode.

pow()	Return power raised to a number.
print()	Print objects to the stream.
property()	Return a property attribute.
range()	Return an iterable sequence.
repr()	Return a string containing a printable representation of an object.
reversed()	Return a reverse iterator.
round()	Return the rounded floating point value.
set()	Return a new set object.
setattr()	Assigns the value to the attribute.
slice()	Return a slice object.
sorted()	Return a new sorted list.
staticmethod()	Return a static method for function.
str()	Return a str version of object.
sum()	Sums the items of an iterable from left to right and returns the total.
super()	Return a proxy object that delegates method calls to a parent or sibling class.
tuple()	Return a tuple
type()	Return the type of an object.
vars()	Return the `__dict__` attribute for a module, class, instance, or any other object.

zip()	Make an iterator that aggregates elements from each of the iterables.
__import__()	This function is invoked by the import statement.

The sample code for this module will go as follows with some added explanation:

- import RPi.GPIO as GPIO

This first line tells the Python interpreter that it will be using a ‘library’ that will tell it how to work with the Raspberry Pi’s GPIO pins. A ‘library’ gives a programming language extra commands that can be used to do something different that it previously did not know how to do.

- import time

Imports the Time library so that we can pause the script later on.

- GPIO.setmode(GPIO.BCM)

Each pin on the Pi has several different names, so you need to tell the program which naming convention is to be used.

- GPIO.setwarnings(False)

This tells Python not to print GPIO warning messages to the screen.

- GPIO.setup(18,GPIO.OUT)

This line tells the Python interpreter that pin 18 is going to be used for outputting information, which means you are going to be able to turn the pin 'on' and 'off'.

- print "LED on"

This line prints some information to the terminal. Telling the program to "print" something post some text to the interpreter you are using, it is a good way to check if the code is working as you properly intended if you get the right print back.

- GPIO.output(18,GPIO.HIGH)

This turns the GPIO pin 'on'. What this actually means is that the pin is made to provide power of 3.3volts. This is enough to turn the LED in our circuit on.

- time.sleep(1)

Pauses the Python program for 1 second

- print "LED off"

This line prints some information to the terminal.

- GPIO.output(18,GPIO.LOW)

This turns the GPIO pin ‘off’, meaning that the pin is no longer supplying any power.

Appendix Section 2: The Stepper Motor0

For the Raspberry Pi, you can program the pins and outputs using the coding language Python. Take note that this code moves the actual motor in a ratio: 1 “step” in the code = 4 “steps” in the motor. The provided Motor has 400 Steps as a full rotation. So further calculations should incorporate this fact. The following code will be broken up as the functions are explained further and what the lines mean.

- import RPi.GPIO as GPIO
- import time

Within python functions work out of pre-stored data packages called “libraries”. These libraries need to be called at the beginning of the page of code to use functions from it later

- GPIO.setmode(GPIO.BCM)

The GPIO used in front of these functions is directly programming the Raspberry Pi. This “setmode” function specifies the numbering system on the Raspberry Pi that the code will follow to assign the different pins with values for the

- enable_pin = 18
- coil_A_1_pin = 4
- coil_A_2_pin = 17
- coil_B_1_pin = 23
- coil_B_2_pin = 24

Each line corresponds to a different pin on the Raspberry Pi, this labels each pin being used i.e: pin 4,17,23,24, from the earlier chart of the Raspberry Pi

- GPIO.setup(enable_pin, GPIO.OUT)
- GPIO.setup(coil_A_1_pin, GPIO.OUT)
- GPIO.setup(coil_A_2_pin, GPIO.OUT)
- GPIO.setup(coil_B_1_pin, GPIO.OUT)
- GPIO.setup(coil_B_2_pin, GPIO.OUT)

The "setup" function maps the pins as output-ing pins so that any commands of the pins are put forth to the breadboard

- GPIO.output(enable_pin, 1)
- def forward(delay, steps):

The function Def (short for define) creates a new function that you can create yourself. The format for this is:

```
def name(variable1,variable2):
        Step 1
        Step 2
```

Naming the function allows you to call upon it when you execute the code, it can hold multiple variables that you may also name that will be used in the equations and executables inside of the new function. For example the previous code "forward" holds the numerical variables "delay" and "steps" which are used inside the function. Be sure to indent any code after the def code line to signal that those functions are within that loop. The indented code pieces are marked by the hollow circles below

- “def” loop.
- for i in range(0, steps):
    - setStep(1, 0, 1, 0)
    - time.sleep(delay)
    - setStep(0, 1, 1, 0)
    - time.sleep(delay)
    - setStep(0, 1, 0, 1)
    - time.sleep(delay)
    - setStep(1, 0, 0, 1)
    - time.sleep(delay)

- def backwards(delay, steps):
- for i in range(0, steps):
    - setStep(1, 0, 0, 1)
    - time.sleep(delay)
    - setStep(0, 1, 0, 1)
    - time.sleep(delay)
    - setStep(0, 1, 1, 0)
    - time.sleep(delay)
    - setStep(1, 0, 1, 0)
    - time.sleep(delay)

- def setStep(w1, w2, w3, w4):
    - GPIO.output(coil_A_1_pin, w1)
    - GPIO.output(coil_A_2_pin, w2)
    - GPIO.output(coil_B_1_pin, w3)
    - GPIO.output(coil_B_2_pin, w4)

- while True:
    - delay = raw_input("Delay between steps (milliseconds)?")
    - steps = raw_input("How many steps forward? ")
    - forward(int(delay) / 1000.0, int(steps))
    - steps = raw_input("How many steps backwards? ")
    - backwards(int(delay) / 1000.0, int(steps))

This experiment will test out the accuracy of the Stepper motor by utilizing a pulley system to test the motor's positioning capabilities. The testing should be simple assembly with some trial and error. Using the platform provided, a shaft,the wheel and a piece of string, a rudimentary pulley system will be assembled to attach to the motor. The assembly instructions goes as such:

Appendix Section 3: The Servo Motor

The servo motor will use the same Python language so some of the functions will be similar to the stepper motor code but there are some new functions that will be explained in Red so it will be easier for you to read and understand. The new function that the servo motor uses is called pwm. PWM or Pulse Width Modulation is a digital signal that moves the servo motor to a specific position. The length of the pulse determines the position of the servo. The Servo works off of "duty cycles" which are measurements of how long the servo motor runs and at what angle the servo motor stops. After receiving the duty cycle the Raspberry Pi will send a pulse to the servo to move it however much the duty cycle dictates.

- import RPi   ## Import GPIO Library.
- import time   ## Import 'time' library for a delay.

These are libraries. similar to the libraries used in the stepper motor code, and these are completely necessary to operate any new functions that you will see afterwards

- GPIO.setmode(GPIO.BOARD)   ## Use BOARD pin numbering.

This line of code allows you to use the same board pin numbering as the picture schematic given in the stepper motor experiment

```
•  GPIO.setup(22, GPIO.OUT)                    ## set output.
•  pwm=GPIO.PWM(22,100)                        ## PWM Frequency
•  pwm.start(5)
•  angle1=10
•  duty1= float(angle1)/10 + 2.5         ## Angle To Duty cycle  Conversion
•
•  angle2=160
•  duty2= float(angle2)/10 + 2.5
•
•  ck=0
•  while ck<=5:
•  pwm.ChangeDutyCycle(duty1)
•  time.sleep(0.8)
•  pwm.ChangeDutyCycle(duty2)
•  time.sleep(0.8)
•  ck=ck+1
•  time.sleep(1)
•  GPIO.cleanup()
```

The code given moves the servo motor 180 degrees. The length of the pulse will determine the position of the servo. The duty cycle is calculated for an angle and the Raspberry Pi sends the signal every 10 milliseconds. The servo motor will move back and forth between the 2 duty cycles and the 2 angles. For this motor test out changing the second angle to see how far the rotation can go and how many full 360 rotations you can do.

Appendix Section 4: The DC Motor

The following code will not include any new functions you have not seen before, but will make use of the variables Motor 1A, 1B, 1E, and the sleep function to pause the code.

```
import RPi.GPIO as GPIO
from time import sleep

GPIO.setmode(GPIO.BOARD)

Motor1A = 16
Motor1B = 18
Motor1E = 22

GPIO.setup(Motor1A,GPIO.OUT)
GPIO.setup(Motor1B,GPIO.OUT)
GPIO.setup(Motor1E,GPIO.OUT)

print "Turning motor on"
GPIO.output(Motor1A,GPIO.HIGH)
GPIO.output(Motor1B,GPIO.LOW)
GPIO.output(Motor1E,GPIO.HIGH)

sleep(2)

print "Stopping motor"
GPIO.output(Motor1E,GPIO.LOW)

GPIO.cleanup()
```

With this given script, the Raspberry Pi is ready to turn the motors. It will turn on some pins, wait two seconds then turn them off again, shown in the remainder of the script. When using the “NTE1749” you can give the DC motors a direction, by turning one side on to turn in one direction, called pin A and vice versa is pin B. To turn the motor on use a pin called “Enable”, labelled E in the test script connected to pin 22. I'll cover this a bit more later. If the motor didn’t turn, double check your wiring or batteries. Debugging and finding out why something doesn’t work can be annoying, but is a useful step in learning something new!

Within the code direction is dictated by the words HIGH and LOW, seen after GPIO. Turning the DC motor a different direction you alter which motors go high or low. The section of the code you want to focus on is:

```
print "Turning motor on"
GPIO.output(Motor1A,GPIO.HIGH)
GPIO.output(Motor1B,GPIO.LOW)
GPIO.output(Motor1E,GPIO.HIGH)
```

Switch around the HIGH and LOW and see which setup turns the motor the opposite direction and save the code for later.

Appendix Section 5: The PIR Motion Sensor

For this sensor you will be using the following example code given:

```
•   import time
•   import RPi.GPIO as io
•   io.setmode(io.BCM)
•
•   pir_pin = 18
•   door_pin = 23  take out door stuff
•
•   io.setup(pir_pin, io.IN)         # activate input
•   io.setup(door_pin, io.IN, pull_up_down=io.PUD_UP)  # activate input with PullUp
•
•   while True:
        o   if io.input(pir_pin):
        o   print("PIR ALARM!")
•   if io.input(door_pin):
        o   print("DOOR ALARM!")
•   time.sleep(0.5)
```

Appendix Section 6: The Ultrasound Motion Sensor

The Ultrasound Motion Sensor requires some new code and some calculations that will be explained section by section. For the Actual input of code into the Python shell just copy and paste all of the dotted lines into the window, making sure each line is separate.

First, declare the libraries that are being used like the previous code.

- import RPi.GPIO as GPIO
- import time
- GPIO.setmode(GPIO.BCM)

Next, we need to name our input and output pins, so that we can refer to it later in our Python code. We'll name our output pin (which triggers the sensor) GPIO 23 [Pin 16] as TRIG, and our input pin (which reads the return signal from the sensor) GPIO 24 [Pin 18] as ECHO.

- TRIG = 23
- ECHO = 24

We'll then print a message to let the user know that distance measurement is in progress. . . .

- print "Distance Measurement In Progress"

Next, set your two GPIO ports as either inputs or outputs as defined previously.

- GPIO.setup(TRIG,GPIO.OUT)
- GPIO.setup(ECHO,GPIO.IN)

Then, ensure that the Trigger pin is set low, and give the sensor a second to settle.

- GPIO.output(TRIG, False)
- print "Waiting For Sensor To Settle"
- time.sleep(2)

The HC-SR04 sensor requires a short 10uS pulse to trigger the module, which will cause the sensor to start the ranging program (8 ultrasound bursts at 40 kHz) in order to obtain an echo response. So, to create our trigger pulse, we set out trigger pin high for 10uS then set it low again.

- *GPIO.output(TRIG, True)*
- *time.sleep(0.00001)*
- *GPIO.output(TRIG, False)*

Now that we’ve sent our pulse signal we need to listen to our input pin, which is connected to ECHO. The sensor sets ECHO to high for the amount of time it takes for the pulse to go and come back, so our code therefore needs to measure the amount of time that the ECHO pin stays high. We use the “while” string to ensure that each signal timestamp is recorded in the correct order.

The time.time() function will record the latest timestamp for a given condition. For example, if a pin goes from low to high, and we’re recording the low condition using the time.time() function, the recorded timestamp will be the latest time at which that pin was low.

Our first step must therefore be to record the last low timestamp for ECHO (pulse_start) e.g. just before the return signal is received and the pin goes high.

- while GPIO.input(ECHO)==0:
  - pulse_start = time.time()

Once a signal is received, the value changes from low (0) to high (1), and the signal will remain high for the duration of the echo pulse. We therefore also need the last high timestamp for ECHO (pulse_end).

- while GPIO.input(ECHO)==1:
  - pulse_end = time.time()

We can now calculate the difference between the two recorded timestamps, and hence the duration of pulse (pulse_duration).

- pulse_duration = pulse_end - pulse_start

With the time it takes for the signal to travel to an object and back again, we can calculate the distance using the following formula.

$$Speed = \frac{Distance}{Time}$$

The speed of sound is variable, depending on what medium it’s travelling through, in addition to the temperature of that medium. However, some clever physicists have calculated the speed of sound at sea level so we’ll take our baseline as the 343m/s. If you’re trying to measure distance through water, this is where you’re falling down – make sure you’re using the right speed of sound!

We also need to divide our time by two because what we’ve calculated above is actually the time it takes for the ultrasonic pulse to travel the distance to the object and back again. We simply want the distance to the object! We can simplify the calculation to be completed in our Python script as follows:

$$34300 = \frac{Distance}{Time/2}$$

$$17150 = \frac{Distance}{Time}$$

$$17150 \times Time = Distance$$

We can plug this calculation into our Python script:

- distance = pulse_duration x 17150

Now we need to round our distance to 2 decimal places (for neatness!)

- distance = round(distance, 2)

Then, we print the distance. The below command will print the word “Distance:” followed by the distance variable, followed by the unit “cm”

- print "Distance:",distance,"cm"

Finally, we clean our GPIO pins to ensure that all inputs/outputs are reset