Chemical Synthesis Automation

by

Cory Helmuth

A Major Qualifying Project

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Bachelor of Science

in Computer Science

by

May 2022

APPROVED:

Smith, Therese

Abstract

The goal of this project is to develop a robot that can autonomously perform chemical reactions under a fume hood. Performing experiments and chemical synthesis can be time consuming and sometimes dangerous for the individual performing such experiments. This device will allow these reactions to be performed safely and without the potential personal harm. This work could become the baseline for chemical engineering research and set new standards for optimizing work in the field.

Contents

List of Figures

Chapter 1

Problem Description

Performing chemical reactions and experiments can be dangerous and time consuming for scientists and researchers in the lab. Developing a robot to automate multi-step experiments under a fume hood without human interaction would improve safety and efficiency for the people involved. Developing a modular solution that allows for any combination of instruments and devices to be used is the most elegant, scalable, and cost effective method of achieving this goal.

Given a list of available instruments and a list of known verbs, this device will be able to be programmed to perform any number of chemical reactions. The device includes equipment to monitor those chemical reactions. This includes a thermocouple which allows the device to control heat generation of chemical reactions using the syringe pumps and modulating the delivery of reactant. This data is important for the chemical synthesis and can be changed based on the acceptable temperature ranges of each given reaction.

The device will contain configuration data that allows it to check if the correct instruments are installed for a given instruction list. The configuration data also serves other potential features, like automatic instruction ordering and error checking for new reactions. State diagrams can be made to visualize the steps the device will take to complete the reactions and show what other possibilities are available.

Chapter 2

Related Work

2.1 Existing Chemical Synthesis Robots

Very few existing off-the-shelf devices accomplish the task that this project sets out to achieve, none with the flexibility nor the price point that is attractive enough for wide scale adoption of this emerging technology. Any news articles about such robots today are about custom made and purpose built robots that are not readily available for purchase.

2.2 How this Project Builds Upon Existing Work

Choosing a platform that is scalable and cost effective that fits the modularity and size requirements consisted of a few different options, but ultimately was decided to be an array of Raspberry Pi B+ units. One unit acts as the "conductor," or the organizational brain of the robot, and the other units each control a single instrument. With this setup, each Raspberry Pi unit will only talk to the conductor unit, while the conductor unit coordinates all of the Raspberry Pi units, which in-turn controls the reaction as a whole. In its current iteration, the instrument Raspberry Pi units in the robot have a RS232 HAT (Hardware Attached on Top), which is an add-on board for the Raspberry Pi that adds a serial port to the Raspberry Pi. The serial port is the interface between the instruments and the Raspberry Pi computer.

The Raspberry Pi units are coded in C and utilize the "socket" library to communicate between the instruments and the conductor. They are connected together through Ethernet RJ45 connectors.

Chapter 3

Overall Design

3.1 Device Structure

The device is made up of three distinct components: the conductor, the controllers, and the instruments.

3.1.1 The Conductor

The first is a central Raspberry Pi named the "conductor". It is responsible for interpreting the information the user has supplied, process it, and translating it into a step by step procedure for the reaction to take place. It will maintain a list of instruments that are connected to the device and be able to assign instructions for each. It will also have a pulse on the current reaction and will be able to stop the procedure if the reaction is not operating as expected or danger conditions are met.

The main method of communication between the conductor and the controllers is through a shared memory buffer. This is advantageous because there does not have to be variable tracking between the conductor and the controllers. There also is no explicit communication word for word between the Raspberry Pis, reducing the amount of potential conflict in the communication process. The conductor simply reads the shared memory and can determine what is happening at that exact moment for all instruments. This also opens the possibilities for further expansion like adding new types of instruments and controllers without changing the entire data

structure between machines. As long as each controller simply reads the commands that are intended for them, the device setup can be modified easily.

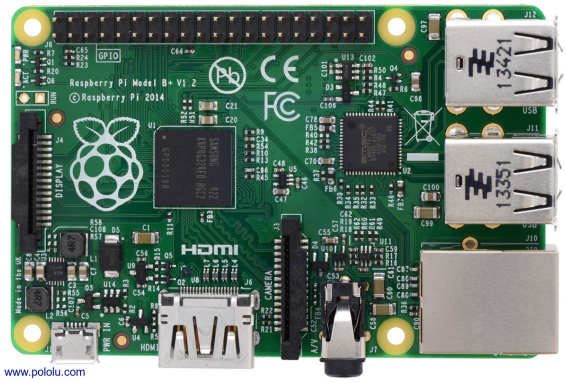

Figure 3.1: Raspberry Pi B+ used for the conductor and the controllers

3.1.2 The Controllers

The controllers are Raspberry Pis which are paired with instruments. Each controller has two processes from using the fork() command. The parent process, or the original process, will communicate with the instrument itself by sending commands and receiving data, when applicable. It can have additional parameters, like data recording, that it takes care of and prepares to be sent to the conductor.

The child process is the process that actually communicates with the conductor. It does this by accessing the shared memory to both receive commands and to send back the requested data.

3.1.3 Instruments

The instruments are the actual devices used to perform the reaction. They are controlled by the controllers and, in the current iteration, use RS232 interfaces. This is a standardized interface that has many resources and existing packages that allow

the controller to communicate with the instruments. It also is widely used among a lot of professional and custom scientific equipment.

3.2 Current Instruments

In the current iteration, the device contains two syringe pumps, a stirrer/hot plate combination device, and a thermocouple. After viewing a list sent by our sponsor AbbVie, the potential final instrument list contains the following:

1. Syringe Pumps
2. Stirrer
3. Hot Plate
4. Bunsen Burner
5. Heating Mantle
6. Thermocouple
7. Conductance Sensor
8. Reactor Effluent Routing Valve

Other instruments—such as devices for isolation of the products or analytical equipment—will be involved with the robot as well, but this MQP will deal primarily with the aforementioned equipment.

3.2.1 Heating

The stirrer and hot plate combo is one of the instruments in the automated system. It is made up of two things, the IKA Plate Magnetic Stirrer and Hot Plate,

Figure 3.2: RS-232 HAT for Raspberry Pi

and a Raspberry Pi B+. The Raspberry Pi has a Serial RS232 HAT attached to it, which adds a serial port that the Raspberry Pi will use to control the IKA Plate. The hot plate has a selection of commands that are available,[IKA] but the project plans on using the following:

1. Set Hot Plate Temperature
2. Turn On Hot Plate

Figure 3.3: IKA Plate, capable of heating and stirring

3. Turn Off Hot Plate
4. Set Stirring Rod Speed
5. Turn On Stirring Rod
6. Turn Off Stirring Rod

The Stirrer Hot Plate Raspberry Pi will listen to the conductor Raspberry Pi for instructions, and send back information when requested. The conductor is in charge

of controlling the instruments, while monitoring for abnormal conditions.

3.3 RS232 Hat

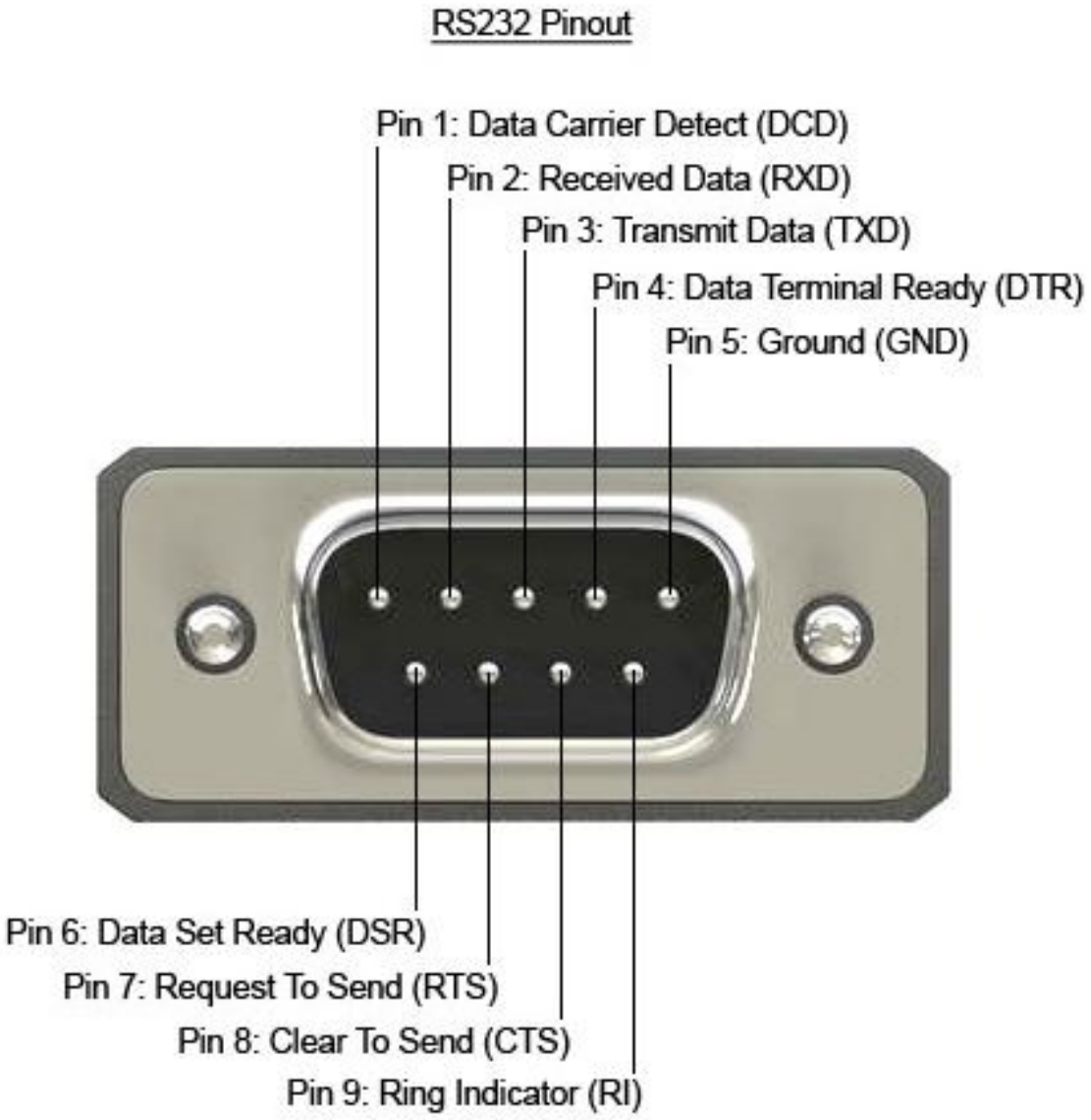

Figure 3.4: RS-232 Pinout

The interface that all of the instruments in the device use is a serial rs232 port.

The device manages this by using a HAT, or a Hardware Attached on Top device for the Raspberry Pi. HATs are widely available and greatly expand the usability of Raspberry Pis, along with widely supported python packages.[Git] This makes it a great choice to use for condensing the device as a whole while keeping the functionality the device needs.[How] It is also convenient because the serial RS232 port is a feature already widely adopted by scientific equipment for controlling devices, meaning most modern scientific equipment won't have trouble integrating with the robot.

3.4 Code

The code written during this MQP is based around the IKA Plate Stirrer and Hot Plate. Based on research about existing packages, the IKA Plate manual, and other implementations of this device in other projects, Python was chosen to communicate with the instrument. The IKA library, developed by Hein Group, served as a guide to implement the required features and be able to send and receive commands and messages. There are also many packages that exist that handle serial communication in Python. These factors are what allowed the code written in Appendix A to be written. Only one example of the scripts is included as they all are approximately the same code with very slight modifications to the strings passed.

The project ended up using the Python package "serial", which gave us the functionality to format and encode the commands before sending. From the main C code, we use the *system*() method to call the script through the terminal, which runs the Python code and exits, allowing the C code to continue. We can pass in arguments based on the command required as well.

3.5 Voltage Booster

A voltage booster is required to boost the output voltage from the Raspberry Pi RS232 HAT. The specification [Fun] for RS 232 Serial ports requires a minimum voltage of 5, both positive and negative, to produce a logical 0 or 1 to the device. The nominal voltage of the TX port on the HAT is about 5.6V, but during transmission the voltage fluctuates anywhere between 4.4 and 5.6, which means that data will be unreadable to the instruments and data will be dropped.

Chapter 4

Results and Discussion

Good work has been put in during this MQP's time, and the work done has pushed the bar forward, closer to the finish line for this project. The code written allows for the main process to easily call the python scripts in order to send commands to the instruments. The list of commands have individual scripts that separate the logic into simple system calls which makes it easier to code and determine the exact input arguments. The voltage boosters will ensure reliability with the signal, and the various checks to ensure proper operation all set a good baseline to complete the project in the future.

There is still work to accomplish before the device is complete and suitable for usage. Once the instruments are operational and able to be controlled by the controller Raspberry Pis, the code will need to be buttoned up and finalized into a more user-friendly application.

In conclusion, this project has made great steps to accomplish the final goals and fulfill the stakeholders requirements. The progress has also shown that the requirements can be accomplished at reasonable cost figures and should be easy enough to allow anybody without prior coding or system experience to use the system.

Chapter 5

Limitation and Future Work

5.1 Finishing the current requirements

The project requires a few things before it can be considered completed. First, the controllers need to be able to send and receive commands and messages from the instruments. After analysis with a logic analyzer, we can determine that messages are correctly being transmitted from the controller to the instrument, but the voltage drops while transmitting and, therefore, cannot be interpreted by the instrument, thus the voltage booster is required. This is basic functionality that should be the number one priority for the next iteration. Without this, it will be extremely hard to debug the next iterations, while also being one of the major stakeholder requirements, which is the ability for the device to perform the reaction steps.

The next steps for the project is to finalize the procedures and code for reading and sharing information over the shared buffer. This would require a system to be developed for communicating to each controller where to read and how much space is allocated to each device.

Lastly, the process for adding instructions manually or using the automatic instruction parser in the conductor needs to be finalized and combined with the shared memory buffer to share the instructions to each controller.

5.2 Additional Instruments and Programming Capabilities

Future work can be done on expending the compatible instruments. In the project's current iteration, the list of devices is rather small. Expanding the available instruments to include dry reactant capabilities and advanced monitoring equipment would expand the use-cases.

As well as adding additional instruments, adding more functionality with the monitoring equipment would expand flexibility. Allowing a conditional instruction based on a measuring tool, for example starting the magnetic stirring rod when the temperature of the reaction reaches a given temperature, would expand the applications the device could be used in.

5.3 Interactive Graphical User Interface

A potential expansion to the existing plan for the device is to develop a fully fleshed out GUI that contains all of the tools required for adding more instrument and controller pairs. This could also include the functionality of adding more types of instruments as well as adding multiples of the same type. This widens the user-base and allows this device to be used by more people and be useful in more applications.

Bibliography

[Fun] Fundamentals of rs-232 communications. https://www.maximintegrated.com/en/design/technical-documents/tutorials/8/83.html.

[Git] Gitlab repository: Hein group / ika. https://gitlab.com/heingroup/ika.

[How] How to use the modmypi serial hat. https://thepihut.com/blogs/raspberry-pi-tutorials/how-to-use-the-modmypi-serial-hat.

[IKA] Ika plate downloads. https://www.ika.com/en/Products-Lab-Eq/Magnetic-Stirrers-Hot-Plate-Lab-Mixer-Stirrer-Blender-csp-188/IKA-Plate-Downloads-cpdl-25004735/.

Appendices

Appendix A

IKA Plate Controller Code

This Python script sends a command to change the motor speed, where it takes in an input argument argv[0], and sends a command to the IKA Plate. The difference between this script and the other five scripts are two things. The following commands are currently in use:

```
START_4     = Start the motor
STOP_4      = Stop the motor
OUT_SP_4    = Set the motor speed
START_1     = Turn on the hot plate
STOP_1      = Turn off the hot plate
OUT_SP_1    = Set the hot plate temperature
```

If the command one of the start or stop commands, the input argument is not used.

The script code for setting the motor speed:

```
import time
import sys
import logging
import threading
import atexit
from datetime import datetime
```

```
try:
    ser = serial.Serial(port="/dev/serial0", baudrate=9600,
    bytesize=serial.SEVENBITS,
    stopbits=serial.STOPBITS_ONE,
    parity=serial.PARITY_EVEN, timeout=0)

except Exception as e:
    print(e)
    print("Failed to connect")

value = sys.argv[0] #Ensure the first argument is the value to set the motor speed to
command = "OUT_SP_4 " + value + "\x0d\x0a" #format the command
print(command)
encoded_msg = command.encode() #encode the message
ser.write(data=encoded_msg) #send the command
print("Command Sent")
```