SIT 210: Embedded Systems Development

Automatic Cup and Drink Dispenser

A finite state machine designed to help those experiencing a loss of fine motor skills

Chloe Hulme

ID: 221007028

TABLE OF CONTENTS:

Background 3
Problem Statement 3
Requirements 3
Design Principals .... 3 - 4
Materials 5
Prototype Architecture: Hardware .... 5 - 6
Prototype Architecture: Software .... 6 - 8
Prototype Code 8
Testing Approach 8
User Manual .... 8 - 9
Proposal for Extension 9
Conclusion 9
References 10

BACKGROUND:

Motor impairment refers to the loss ones fine motor skills as a result of disability or disease. A study conducted in NSW determined that the frequency of motor impairment among adults dramatically increases with age. Around 21% of adults aged 55 and above experience one form of motor impairment, with this figure increasing to 42% in adults over 85 [1]. The loss of one's fine motor skills has the potential to impact an individual's independence, which in turn can impact an individual's mental health and well being. It is proven that experiencing disorders such as Cerebral Palsy and Motor Neurone Disease - both of which deteriorate fine motor skills, can have a severely negative impact on individuals' mental health. Often leading to higher instances of depression and anxiety [2][3]. Which in turn, can lead to a diminished sense of self, and a feeling of loss of control in one's life.

PROBLEM STATEMENT:

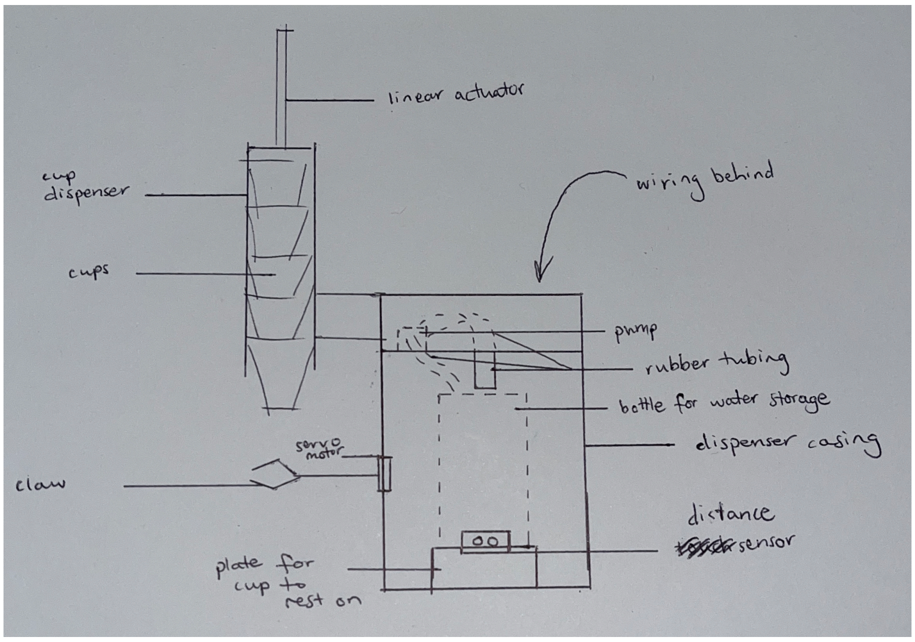
Currently, automated drink dispensers tend to focus on the dispensing of liquid, and there are few accessible systems that are also able to dispense a cup. However, for an individual experiencing a loss of fine motor skills, hand muscles are often impaired. Therefore, fetching a cup can be extremely difficult. This system addresses this by dispensing a cup for the user. The goal behind this system is to allow an individual experiencing motor impairment to regain a level of independence through providing assistance in completing a simple task. The development of a system that is able to dispense both a cup and water has the potential to do so; and thus this system can help those experiencing a loss of fine motor skills to exercise a greater level of self-determination and self-reliance in their daily lives. The prevalence of which, is able boost mental health and wellbeing, and quality of life. Additionally, drink dispensers will often have a touch screen interface, which is not conducive to use by a motor impaired individual. This system addresses this by employing use of voice activation, and a joystick remote (which is a common remote type already used by those experiencing motor impairment, eg. On an electric wheelchair; and is highly intuitive for most users). The user will also be provided with the option to interact with a GUI, should this be their preference.

REQUIREMENTS:

This system must efficiently and effectively deliver a cup beneath a liquid dispenser, then return a liquid filled cup to the user. Given that individuals experiencing a loss of fine motor skills may require mobility aids, they may move relatively slowly. Hence, this system must not attempt to reset until it is confirmed that the user has retrieved their drink. Additionally, this system must provide users with multiple user interfaces that do not require use of fine motor skills, and that do not require direct contact with the system itself. In this implementation, voice activation is a viable option, however those experiencing degenerative disabilities will need additional options as their condition progresses. Hence, employing use of a joystick remote control is also essential. Moreover, this system must not attempt to dispense multiple drinks at a time. That is, the machine must fully dispense and reset before another drink is able to be dispensed. This is due to the potential for accidental knocking of the remote. Finally, the system must be able to be easily reset and reloaded with cups.

DESIGN PRINCIPALS:

The prototype design is best described using a visual aid. Below is the technical drawing of how each component of the system is fitted together.



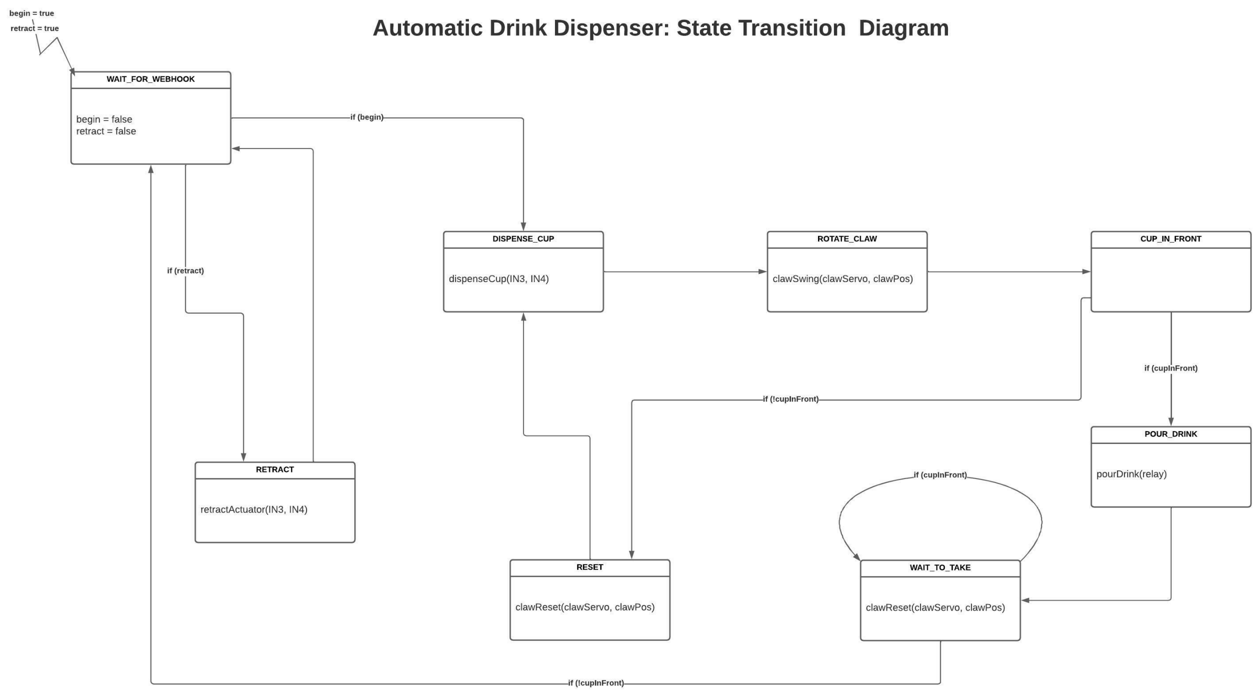
Evidently, we have the body of the system which fixes each component in place, and encases the wiring of the system. We have the cup holder/dispenser to the left, with a linear actuator fitted on top to drive the cups through the cup dispenser. Below this is a claw attached to a servo motor, its purpose being to catch the cup once it is released from the dispenser. This servo will swing the claw and cup around in front of an ultrasonic distance sensor, which tests whether the cup is in the correct position. Once the cups position is verified, the single channel relay (not pictured in the technical drawing) will activate the pump, and water will begin to pour into the cup. Once the cup has been retrieved, the claw position resets for the machines next use. The machine can be operated by interacting with a bluetooth-capable joystick remote controller, a graphical user interface (GUI), or with voice activation using a Google Assistant.

The overarching design of the system takes the form of a finite state machine. Meaning, the system has a limited number of conditional states of being. The state transition diagram below provides a visual aid of the flow of operation the system transitions through when in use. The benefit of this design is four-fold. Firstly, the system begins in, and can maintain an idle state. This is highly practical given that if this system were to be developed beyond the prototyping phase, users would be able to leave the machine powered on at all times, with the machine drawing limited power. For example, the way a coffee machine or a toaster is always on, and in an idle state until interacted with. This therefore supports ease of operation by eliminating the need to setup/power-on the machine before each uses; this also means the machine will not begin performing any tasks until it is interacted with and is therefore cost effective for users as the machine is not constantly operating. Moreover, this supports ease of use specifically for those experiencing a loss of fine motor skills, as they may find it especially difficult to set up a machine before each use, if it was necessary to do so.

Secondly, this system design incorporates a level of fault tolerance. This is most clear in the inner loop of the state transition diagram: If the ultrasonic distance sensor does not detect a cup has been caught by the claw, the machine will reset and attempt to dispense a cup once more. The machine will repeat this until it is detected that there is indeed a cup in the correct position. Once this is verified, the rest of the machines operation will take place. This is beneficial as it ensures a drink will not be poured without a cup in place, limiting spillage. This also ensures that the system will inevitably fulfil its desired purpose each time it is operated, meaning the user will always get their drink - even if it takes a couple of tries to dispense the cup.

Thirdly, this design is beneficial specifically to those who may be less mobile or using mobility aids that slow them down, as the machine will not reset itself until it is verified that the cup has been retrieved by the user. The machine will loop through a single state until the ultrasonic sensor can verify the cup has been retrieved by the user. Ensuring the user is able to take as much time as necessary to retrieve the cup.

Finally, by designing the system as a finite state machine, we are able to build in a state that allows the machine to be fully reset so cups can be reloaded into the machine. This is obviously useful for when the machine used all of the cups initially loaded into the cup dispenser.



MATERIALS:

- Paper cup dispenser
- Linear actuator (home made using DC motor, L298N dual H-bridge motor driver, long headless screw and nut, paper, and super glue)
- Servo motor
- Claw (made of popsicle sticks)
- Ultrasonic sensor (H-SR04)
- Micro air pump
- Single channel relay
- Battery pack
- Rubber tubing
- Bottle to store liquid
- Materials to build casing
- Joystick
- Arduino nano
- Bluetooth module (HC-05)
- Particle Argon
- Raspberry Pi 3 Model B+

PROTOTYPE ARCHITECTURE: HARDWARE:

The hardware design choices were made specifically to maximise energy efficiency, and conform to certain design constraints of the end product and the problem domain. The system has therefore been designed assuming it was to be developed beyond the prototyping phase.

Firstly, the most pertinent choice to address is what each system embedded within the machine is responsible for and why. There are three systems in operation that work together to perform the function of the system as a whole. They are run on Raspberry Pi (RPi), Particle Argon, and an Arduino Nano. The Raspberry Pi is responsible for operating the ultrasonic distance sensor, the graphical user interface (GUI), and the receiver side of the joystick remote controller. The Particle Argon is responsible for overseeing the function of the machine and therefore runs the finite state machine; while the Arduino Nano is responsible for the sender side of the joystick remote controller. Given that the Particle Argon and Arduino Nano are not capable of running multiple programs in parallel, the RPi must take on the responsibility of managing multiple aspects of the system. In addition, given that the finite state machine is run on the Particle Argon, we use the RPi to oversee the user interfaces and provide the entry point to initialise the transition out of the idle state into the next state. Moreover, the RPi operates the ultrasonic distance sensor as the Particle Argon is incapable of interfacing with an H-SR04 ultrasonic distance sensor, and the Arduino Nano must be wireless to the machine. Similarly, the Particle Argon used in the system is not bluetooth capable, and therefore the RPi must be responsible for the receiver side of the joystick remote controller.

Secondly, an HC-05 bluetooth module has been chosen to provide bluetooth communication between the RPi and the Arduino Nano as this module is affordable and provides traditional bluetooth communication rather than BLE. While BLE is more energy efficient than classic Bluetooth, interfacing the RPi with BLE modules such as the HM-10 proved very difficult, and there was limited supporting documentation on this concept available. The HC-05 communicates with the Arduino Nano using the UART communication protocol. This is appropriate given we only require these two devices to be involved in this communication, and hardware complexity of this communication protocol is low [4]; which is beneficial as we are aiming to keep the remote controller as compact as possible. Additionally, Bluetooth was chosen as the communication protocol between the RPi and the Arduino Nano primarily because it is wireless and has reasonable range, with the RPi comfortably supporting the Radio Frequency Communication (RFCOMM) bluetooth protocol as the receiver ('host') side of the communication [5]. Further to this, we need the remote controller to be as compact as possible as it must suit the ultimate goal of being able to be velcro strapped to a wheelchair. The remote controller therefore uses the Arduino Nano as its microcontroller due to its size. The remote controller would sit inside a 3D printed casing, with the joystick exposed for use. The joystick itself has been chosen as it has highly intuitive interface, and is operated the same as devices already widely used by those experience a loss of fine motor skills, such as motorised wheel chairs. Therefore it is reasonable to assume a joystick provides ease of use for such individuals.

Finally, as linear actuators are quite expensive, a homemade linear actuator was used in the prototype of the system. This involved attaching a headless screw and nut, wrapped in paper (weighed down with additional nuts) to a DC motor. This motor is driven by a L298N dual H-bridge motor driver and operates using Pulse Width Modulation (PWM). PWM was used here as it improves the efficiency of the motor. With PWM, we are able to control the amount of power delivered across the motor without dissipating any wasted power [6]. This therefore improves the overall efficiency of the system, which is especially beneficial as there are multiple sensors/ devices connected to the system.

PROTOTYPE ARCHITECTURE: SOFTWARE:

The software design choices are based around the hardware design constraints, as mentioned above, as well as the premise of designing the machine as a finite state machine. Therefore we will discuss the software architecture on a system-by-system basis and describe how each system is connected.

Firstly, on the Particle Argon, we are running the finite state machine. It was determined that the best way to organise the file structure on the Particle Argon was to run the switch statement, state enumeration and all global variables/functions relating to the finite state machine in the .ino file provided by the Particle Web IDE. We then have two layers of abstraction below this primary file that are .cpp files and their corresponding header file for a total of 3 layers of abstraction within the machine software. Thus, below the primary .ino file, we have a .cpp file with its header file that is responsible for all functions relating to the individual components in the machine, ie. the servo motor, the pump, the DC motor etc.. Below this file is a .cpp file with its header file that is responsible for all functions relating to the linear actuator, ie. retracting, extending and stopping. Using this model of abstraction, a file is only exposed to the file directly above itself. Therefore, the primary .ino file does not include the header file relating to the linear actuator, however the first .cpp file consisting of component functions does. Thus the code running on the Particle Argon is neatly organised, highly readable, easy to extend and easy to debug. Additionally, the transition into the first state after the idle state runs due to a series of global variables that act as switches. For example, the global variable 'begin' is initially set to False. Once this variable is set to True, the finite state machine transitions out of its idle state and begins operation. After operation has begun, this global variable is set back to False in preparation for the machines next use. All code running on the Particle Argon is written in C++.

Secondly, the programs running on the RPi provide the entry code to the finite state machine. They are relatively simple programs, we will run through each of them separately. The first program runs the GUI using the tkinter library. This program simply creates 3 buttons and attaches functions to each button. There is the option to initiate machine operation using the 'Water!' button. Users are also able to retract the linear actuator using the 'Retract' button. The exit button will close the GUI. When a user presses the 'Water!' or 'Retract' buttons, a webhook that contains a unique key generated by If This Then That (IFTTT) is posted. The corresponding IFTTT applet uses this webhook to then publish a message to Particle. These messages have been subscribed to in the Particle Argon code, and once the Argon receives notice that the publish message has been received, the Argon will run a corresponding function. To aid clarity, we will run through this step-by-step:

1. The User interacts with the graphical interface, pressing the 'Water!' button.

2. A webhook containing a unique key is posted.

3. This webhook is captured by IFTTT, the unique key is identified as belonging to the applet that publishes the message 'water' to Particle.

4. The 'water' message is published to Particle.

5. The primary .ino file on the Particle Argon subscribes to the 'water' message in setup. Once this message is published by Particle, the subscribe statement (that is subscribed to 'water') runs its event handler.

6. The event handler sets the global variable 'begin' to True and the machine begins operation.

The joystick remote controller on the RPi works in much the same way. However, it relies on receiving values within a specific range from the HC-05 Bluetooth module, to determine which webhook is posted. Values above 900 will post the webhook relating to dispensing a cup and water, while values below 100 will post the webhook relating to retracting the linear actuator. Finally, on the RPi, we have the ultrasonic distance sensor. This will detect whether there is a solid object within 10cm in front of the sensor. Similar to the other programs running on the RPi, If the cup is/is not in front of the sensor, a webhook will be posted. The corresponding IFTTT applet uses this webhook to publish the Particle message, which has been subscribed to on the Argon. . Once this message is published, the subscribe statement runs its event handler. Likewise as with the 'begin' global variable, the global variable that states whether a cup is/is not in the correct position is set to True/False. This will determine whether the machine needs to try retrieving a cup again, or whether the user has taken the cup from the machine so the machine can reset itself. All code running on the RPi has been written in Python.

The program running not the Arduino Nano is very simple. The joystick is an analog sensor that will output a value determined by the direction it is moved in. If the joystick is moved far left, values above 900 are generated, whereas if the joystick is moved far right, values below 100 are generated. Values with in the range $(\infty, 900) \cup (100, 0)$ are then sent via bluetooth to the RPi, the corresponding joystick program on the RPi then handles the rest of this process. All code running on the Arduino Nano is written in C++.

Finally, users can interact with their Google Assistant to operate the machine as well. This is simply an IFTTT applet that uses the Google Assistant as a trigger, with the event being to publish the particle message ‘water’. This runs the exact same was as described above (from step 4 onwards).

PROTOTYPE CODE:

Link to prototype code: https://github.com/chloeehulme/ES_ProjectSubmission.git

TESTING APPROACH:

The primary testing phase of the prototype involved writing test programs for each individual component of the machine, and testing each component in isolation. First the liner actuator was built and tested. A simple program was built that set the duty cycle of the DC motor to 255 (the maximum) just to test whether the home made linear actuator was actually functional. Once it was determined that it could be used to drive cups through the dispenser, it then had to be determined how long the linear actuator had to be extended before a cup was dispensed. This was done by extending the linear actuator for an arbitrarily long amount of time, and eventually landing on 10 seconds. Then it was determined in a similar fashion that it takes around 30 seconds to fully retract the linear actuator back to its initial position. The next component tested was the servo motor. A claw was fashioned out of popsicle sticks and glued to the servo. A simple test program was then built to determine the angle the servo needed to rotate to until the cup had been successfully moved from under the cup dispenser to under where the water dispenser would be. It was found the servo must rotate 180°. After this, the single channel relay and pump were tested using a basic testing program. All that was tested was simply turning the pump on and off, and testing how long to run the pump until a cup was filled with an adequate amount of water.

Next, these individual components were tested in succession in a test program, with the goal of fine tuning the running of the entire machine as a whole. Once the machine ran smoothly, and a cup could be successfully dispensed, with water being poured into the cup, the testing of the RPi programs began. There was limited testing involved in the GUI itself other than verifying each button is displayed correctly. However, dummy IFTTT applets were made to verify that each webhook is posted and detected by IFTTT correctly. After this, the joystick value range was tested using a test script on the Arduino Nano, and then the HC-05 and RPi were paired [7][8]. Successful pairing was verified by sending test values from the Nano to the RPi. The webhooks on the RPi side of this program need not be tested as they are the exact same as in the GUI program. Finally, the distance sensor was tested and it was determined the appropriate value that will verify a cup is situated correctly under the water dispenser is 10cm. The webhooks in this program were tested the same way as in the GUI program.

Once all the preliminary testing had been completed, the code on the Particle Argon was restructured into a finite state machine, and final testing of the system took place. This involved ensuring the user interfaces correctly initialised various machines operations. Finally, the applet connecting the Google Assistant was created and tested.

USER MANUAL:

1. Load paper cups into the cup dispenser by lifting the plastic cover off the top of the cup dispenser ad inserting cups. Replace the plastic cover once complete.

2. Fill the jug with water and replace behind the machine.

3. Power on the machine by connecting the RPi to a wall socket and switching it on, then connecting the Particle Argon to an external power source (wall socket with an appropriate adapter, or laptop).

4. Once the machine is powered on, wait until the LED on the Particle Argon stops rapidly blinking.
5. Open the RPi terminal using remote connection to the RPi, and type the following command: *sudo rfcomm connect hci0 <00:00:00:00:00:00>*, inserting the MAC address of the HC-05 module where the brackets are.

6. Verify connection by running the following command in a seperate terminal window:
cat /dev/rfcomm0

7. Begin running the RPi programs by typing the following command into terminal:
sudo python bluetooth_joystick.py & python dispenserGUI.py & ultrasonic_sensor.py

8. Check that the GUI has loaded correctly.

9. Operate the machine using your desired interface.

10. Once the machine has run out of cups, retract the linear actuator using the joystick or GUI, then refill cups as per step 1.

PROPOSAL FOR EXTENSION:

If this machine was to be extended beyond this prototype, the following extensions should be considered. Firstly, installing a liquid level sensor in the water jug, and creating an IFTTT applet that pushes a notification to your smart device once the liquid level is considered low. This would let the user know the water needs to be refiled without them having to manually check. To take this a step further, IFTTT allows you to add to your google shopping list. If a user has drinks other than water, ie. juice, milk etc., the applet would add these beverages to your shopping list automatically. Further to this, refrigeration should be considered a highly necessary extension for health and safety, particularly if users are going to use this machine for beverages such as milk. Conversely, the ability to pour heated/boiling water would be a fantastic extension as well as it would enable more complex drinks such as coffee to be made. Therefore the machine would need to be able to dispense multiple drinks (and also solids such as instant coffee or tea bags) based on user selection.

CONCLUSION:

The experience of taking this prototype from an idea, to a functioning system has been both challenging and rewarding. The greatest takeaway from this experience has definitely been to plan thoroughly, but also maintain a level of flexibility as things inevitably go wrong and we must deviate from the path we had laid out for ourselves. The most challenging aspect of creating this prototype was successfully operating a home made linear actuator. This experience highlighted how important perseverance and maintaining presence of mind under time constraints can be.

Given the opportunity to redo this project, the following changes would be made. The claw attached to the servo motor would have been designed in Fusion 360 to better fit the shape of the paper cup, then 3D printed. This is due to the fact that the cup keeps slipping out of the poorly shaped claw. Additionally, a more pleasant looking casing would be built for the machine, this would help the prototype look a bit more professional, and perhaps even arrange the components in a better way to make the machine more compact. Moreover, a linear actuator would be purchased as, despite how expensive they are, having an electric linear actuator would have significantly reduced the amount of time and frustration spent trying to get the homemade linear actuator to function properly. Finally, users would have been given the option of selecting between two drinks. This would have demonstrated the scalability of the prototype.

REFERENCES:

1. Herbert, R.D., Taylor, J.L., Lord, S.R. et al., (2020). *Prevalence of motor impairment in residents of New South Wales, Australia aged 55 years and over: cross-sectional survey of the 45 and Up cohort.* [Website]. Available: https://doi.org/10.1186/s12889-020-09443-5

2. Beier M. L., (2022). *Multiple Sclerosis and Mental Health: 3 Common Challenges.* [Website] Available: https://www.hopkinsmedicine.org/health/conditions-and-diseases/multiple-sclerosis-ms/multiple-sclerosis-and-mental-health-3-common-challenges

3. Hogg KE, Goldstein LH, Leigh PN, (1994, August 24). *The psychological impact of motor neurone disease*. [Website]. Available: https://pubmed.ncbi.nlm.nih.gov/7991744/

4. RF Wireless World, (2012). *Advantages of UART | disadvantages of UART.* [Website]. Available: https://www.rfwireless-world.com/Terminology/Advantages-and-Disadvantages-of-UART.html

5. Wikipedia, (2022, May 5). *List of Bluetooth protocols.* [Website]. Available: https://en.wikipedia.org/wiki/List_of_Bluetooth_protocols

6. Electronics Tutorials, (2022). *Pulse Width Modulation.* [Website]. Available: https://www.electronics-tutorials.ws/blog/pulse-width-modulation.html

7. NotQuiteHere, (2022, April 26). *Connect an Arduino to a Rasberry Pi via an HC-05 Bluetooth Module.* [Website]. Available: https://tamarisk.it/connect-an-arduino-to-a-rasberry-pi-via-an-hc-05-bluetooth-chip/

8. Joshi M., (2019, July 21). *Raspberry Pi 3 and Arduino communication via bluetooth HC-05.* [Website]. Available: https://medium.com/@mahesh_joshi/raspberry-pi-3-and-arduino-communication-via-bluetooth-hc-05-5d7b6f162ab3