The IoT Smart Pet Feeder

CITS5506, Semester 2 2020

Jordan Harun, Master of Software Engineering

Tom Nicholls, Master of Data Science

Farhad Ahmed, Master of Electronics Engineering

Clariza Look, Master of Information Technology

Abstract

The IoT pet feeder is a project looking at integrating internet connectivity to an everyday task. The design was constructed to allow for remote control access, through the Internet of Things, to be able to feed your pets whilst not physically on the property. As pet obesity and irregular feeding for pets has been observed to be prevalent in modern society, the IoT pet feeder provides a solution to the troubles in this domain. After extensive research, the team settled on a design based on a Raspberry Pi 3 (minicomputer) and Blynk, an IoT platform. Linking these two together allowed the team to create an IoT based system, which allows for a feeding mechanism to dispense food to pets, specifically chickens. With a combination of hardware and software collaboration, the team was able to successfully integrate a sensor system to detect chicken movement, as well as deploy a camera for a live feed at the chosen premise. The IoT pet feeder can be programmed to dispense feed at intervals throughout the day, once detected by the sensor on the IoT device or remote controlled by the user linked with the device. The IoT pet feeder that was created was successful, albeit, needs some adjustments if it were ever to go into production. Modifications to improve the IoT pet feeder includes improving functionality, integration to a home system, mechanism to allow for greater variety in feed types and computer vision capabilities.

Keywords: *IoT (Internet of Things), pet feeder, pet obesity, Raspberry Pi, Blynk, servo motor*

1. Introduction

The introduction of new wireless technologies has changed the way humans interact, none more so than the smartphone. The smartphone has become a popular wireless technology because of its diverse range of applications and uses. The smartphone has the capacity to change people's lives, one of the emerging technologies in this regard for smartphones is called the Internet of Things (IoT). IoT is basically referring to any electronic device implanted in functional objects such as wearables or sensors that has the capacity to send and receive information about its environment wirelessly [1]. What makes this new paradigm more exciting is not just its ability to transfer data wirelessly but also the feasibility to implement the technology. In addition to this, IoT devices have the capability to gather massive amount of data that can be used for insights [2]. The project group looked to explore what IoT means and how it could be

integrated into people's lives. As IoT can be stretched over a broad domain, the opportunities the team was presented with were abundant, but the team was enthusiastic to attempt creating an IoT Pet Feeder.

In the recent years, pet adoption has become popular in many households. According to the Mondor Intelligence report in 2019, animal care has become a part of human lifestyle and there is an increasing upward trend in the demand for pet personal products particularly in the pet food market [3]. The study also estimated that there will be a 2.7% increase in compound annual growth rate (CAGR) for the forecasted 2020-2025 period [3]. A survey done by American Animal Hospital Association (AAHA) in the year 2000 showed a high amount of pet owners feeling guilty for putting their dogs in homes while away for work or holidays [4]. In addition to this, their seems to also be an uptick trend in obesity of pets. A study done by the British Veterinary Nursing Association (BVNA) has seen that veterinarians estimated about 34% of feline pets they see per week are considered overweight [5]. Many pet owners are unable to remain at home the entire day to serve their pet; therefore, it is hard to maintain consistency in maintaining their feeding schedules. One of the popular ways to tackle this problem is to place bulk feed in the pet bowl and let the animal control the eating. This method can lead to overfeeding of the pet, which can lead to creating bad habits or inducing health problems. With these problems in view, this project aims to fill in the gap of managing pet's feeding schedule for owners with the use of an IoT technology. The team proposed a pet feeding system that can be managed through a mobile phone wirelessly while the owner is away from home.

This study's underlying motivation is to create a pet feeder system that has the ability to self-serve the right amount of food for the pet at a set time, monitor food allocations and provide an alert system to the pet owner through the mobile phone application. This mobile phone application can also dispense food on demand from a remote location. To achieve this, this project intends to use a Raspberry Pi as the central microcontroller of the feeder (instead of Arduino) and Blynk as the mobile application development platform to control the feed system to the server.

2. Former Related Researches

Over recent years, there have been a few studies done on IoT-based pet feeding system such as the smart dog feeder design with the use of MQTT and Android [7]. Although they address to provide automation to the dog feeding system via the mobile phone, they have used a different hardware system such as Arduino Uno microcontroller, thus requiring a Wi-Fi-module to connect to the server. Their smart feeder is comprised of the parts mentioned above as well as a weighing module, buzzer, RFID reader module, servo motor, Paho Android Service to connect to the server and uses MQTT protocol - a software application that enables IoT-devices to send and receive messages to a web server [7]. The feeding system has a capacity to feed around 1 kilograms of dog food and its design is for small to medium sized dogs.

Meanwhile, our group's smart pet food dispenser is intending to feed small to medium size animals such as cats, small dogs, and chickens to mention a few. It has a camera that provides live video feed to the mobile application that can be viewed by the pet owner. Additionally, it also comes with the motion sensor in which it helps the pet owner identify if the pet is near the smart feeder. The pet food dispensing system is controlled through a Raspberry Pi unit with a built-in Wi-Fi module and uses a Blynk application in the mobile phone.

3. General System Design

Our group's prototype of an IOT-based pet feeding system is composed of units including:

- 1 Raspberry Pi micro-controller,
- 1 Servo motor,
- 1 Motion sensor,
- 1 Raspberry Pi camera and
- A Blynk application as the server and smartphone application interface.

The smart pet feeder has one layer to dispense pet food. Each feeding is about 50 grams of dry pellet that will be dropped to the feeding bowl. The Raspberry Pi acts as the central controller of the servo unit, motion sensor and the camera while it is operated through the mobile application. The camera provides the user the visual of the feeder outwards, where the pet feeding area can be seen through the Blynk mobile application. The user can control the feeding through the Blynk app on the mobile phone where the app is designed to be able to set feeding times, activate alerts if feed controller is empty, and updating the feeding times.

3.1. Microcontroller

Figure 1 below shows the block diagram for the smart pet feeder in which the Raspberry Pi has both the GPIO and Wi-Fi modules. The GPIO is where the motion sensor and the servo motor are connected. The Pi camera is connected to the camera port of the Raspberry Pi. Aside being as the main controller of the entire feeding system, the Raspberry Pi has this Wi-Fi module which enables it to connect to the internet for communication with the Blynk App server. The I/O and status data is synced with the Blynk server while the camera feed is transmitted as a live stream. The microcontroller is connected to a regular home power source and gets its internet connection through a home Wi-fi connection. The Raspberry Pi also acts as the private local server of the entire feeding system.

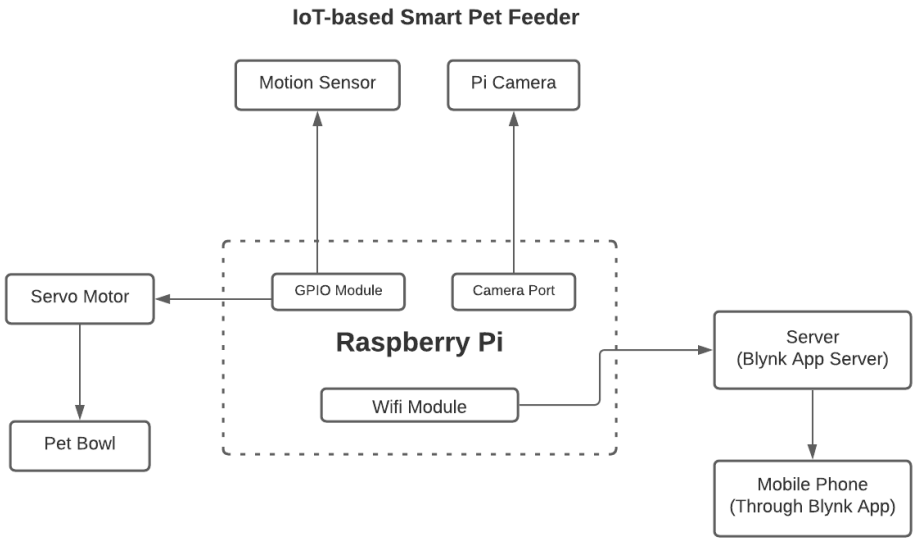


Figure 1: Block Diagram of the Smart Pet Feeder

3.2. Server

The Blynk app server is an open source platform that can be accessed through a mobile application on both IOS and Android, to manipulate microcontrollers such as Raspberry Pi and Arduino [6]. It can operate the Raspberry Pi's hardware as well as any software, whilst pushing data to the Blynk mobile application for monitoring. The Raspberry Pi's wi-fi module enables the micro-controller to connect to the server. It is setup using an integration of Python and C++ code chunks. Since we are using a free account of the Blynk server, we are limited to create only a few features to control the entire feeding system. In order to address that issue, a locally hosted instance would need to be set up, granting unlimited access. For the purposes of this project

3.3. Blynk Mobile App

The team created the mobile dashboard from the Blynk mobile app which is connected to the Blynk server. The Blynk mobile app was downloaded on an Android phone and an account was created to be able to connect to the server. The interface was customised to show the status data as per the requirements. A button is available for manual feed which overrides any timed feed and immediately dispenses the feed on being pressed. The app can also subscribe to the video camera stream which is set up by the Raspberry Pi.

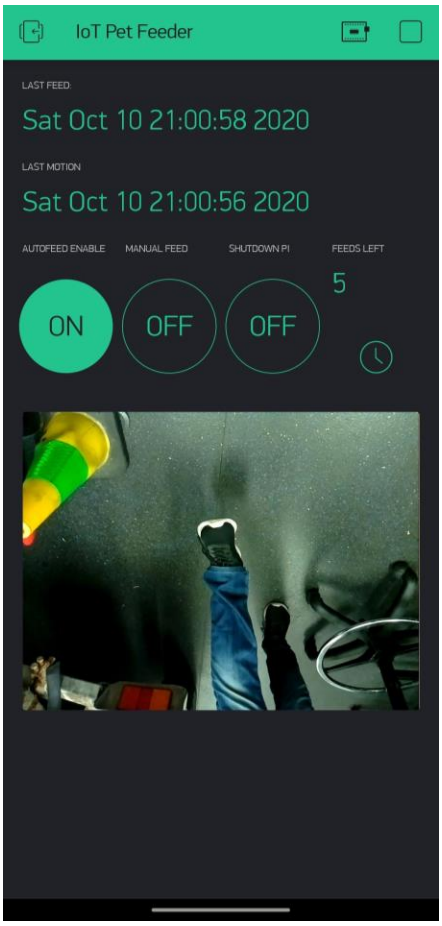


Figure 2: Blynk Mobile App Dashboard

3.4. Servo Motor

The Servo motor in figure 3 has a dispenser mechanism that rotates the 3D printed servo case in Figure 4 to pour the dry pet pellets. However due to the small size servo motor and the servo case composition is made from soft plastic, the weight of the serving size is limited. The angular position of the servo can be varied by varying the duty cycle of the PWM signal fed into it.

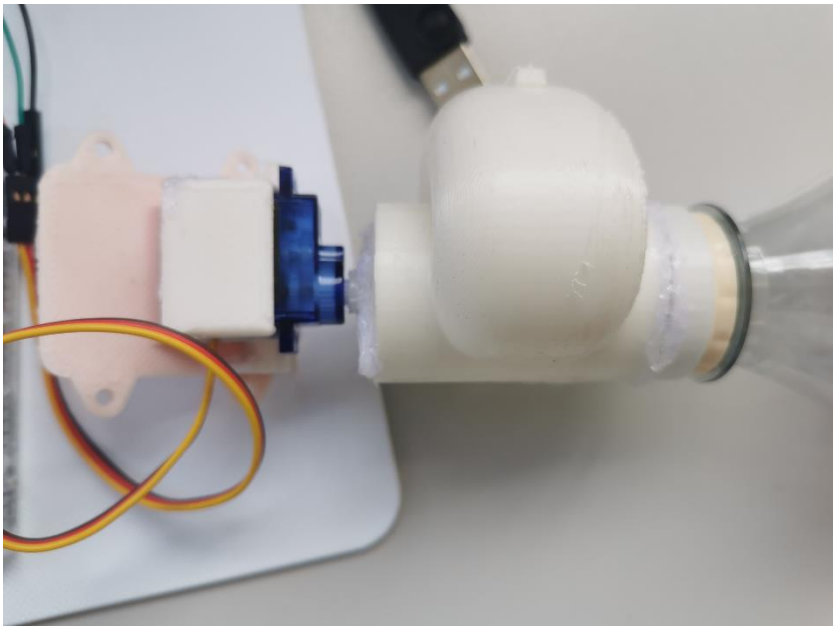


Figure 3: Plastic servo motor attached to the 3D servo case

Figure 4: The 3D model of the servo cased that was used to print

3.5. Pi Camera

The Pi camera in figure 5 is used to have a live visual of the pet to the mobile phone application of the owner. The Blynk mobile app that is installed on the smart phone of the owner has the ability of provide live video feed of the pet from the camera. The camera also provides visuals if the bowl is empty or not.

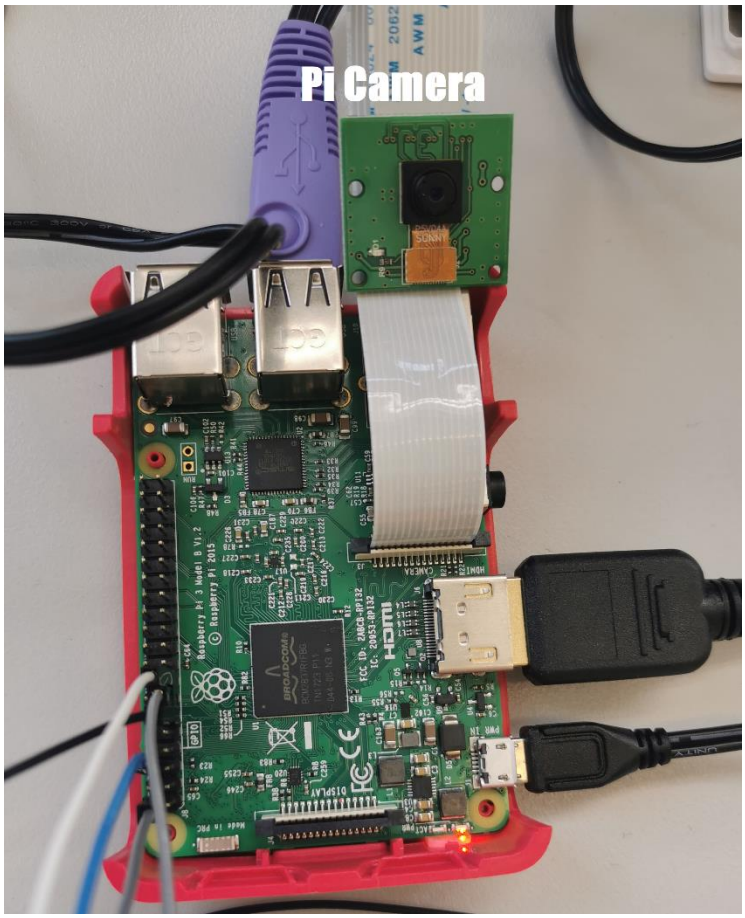


Figure 5: Pi Camera mounted to the Raspberry Pi

3.6. Motion Sensor

The motion sensor is used to detect motion if the pet is near the feeding system. It works by generating a rising edge on its output when a motion starts and a falling edge when motion ceases. Once the controller receives motion signals, it dispenses the food if the time limitations allow for it and updates the status through to the Blynk mobile app. A 3D case was printed for the motion sensor protector so that it will be shielded from responses to any other motion.

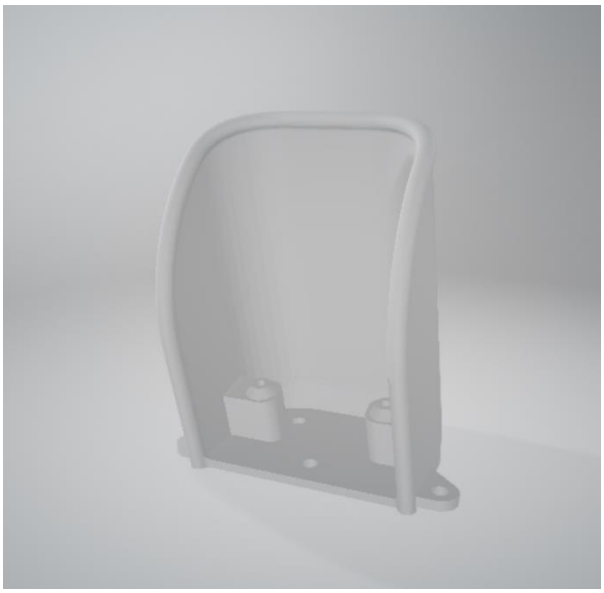


Figure 6: 3D print case model for motion sensor

4. Costing of Components

Item Description	Cost	Web Address Source
Raspberry Pi	AU $60	*https://core-electronics.com.au/raspberry-pi-3-model-b.html*
Breadboard	AU $6	*https://core-electronics.com.au/solderless-breadboard-300-tie-points-zy-60.html*
Wires	AU $9	*https://www.jaycar.com.au/breadboard-jumper-kit/p/PB8850*
Raspberry Pi Camera	AU $39	*https://core-electronics.com.au/search/?q=pi+camera*
Servo Motor	AU $12	*https://www.jaycar.com.au/arduino-compatible-9g-micro-servo-motor/p/YM2758*
PIR Motion Sensor	AU $6	*https://www.jaycar.com.au/arduino-compatible-pir-motion-detector-module/p/XC4444*
TOTAL	**AU $132**	

5. Implementation Choices

5.1. Hardware Architecture Implementation

As already mentioned, Raspberry Pi was selected as the controller for this project. Figure 7 shows the connections available on the controller and Figure 8 shows how the prototype of the group's IoT Pet Feeder has been set up on it.

5.1.1. Electronic Connections

The General-Purpose Input / Output (GPIO) pins on the Raspberry Pi, which are detailed in Figure 9, have been used to connect to the motion sensor and the servo motor. Pin 2 provides 5V and pin 6 provides Ground reference to both devices. Pin 13 (GPIO27) has been configured as a digital output and is connected to the servo motor. The software on the Pi sends PWM waves through this pin which determines the angular position of the servo motor. Pin 15 (GPIO22) has been configured as a digital input and is connected to the motion sensor. Whenever motion is detected, a rising edge is generated at this pin, which is then used by the software to make decisions and take further action. These data pins can be replaced by any other GPIO pin if required.

The Pi Camera is connected to the CSI Camera port to monitor the surroundings of the pet feeder. A live stream is set up using standard libraries.

The Pi is powered through the Micro-USB power input port using the supplied adapter to connect directly to a wall power outlet.

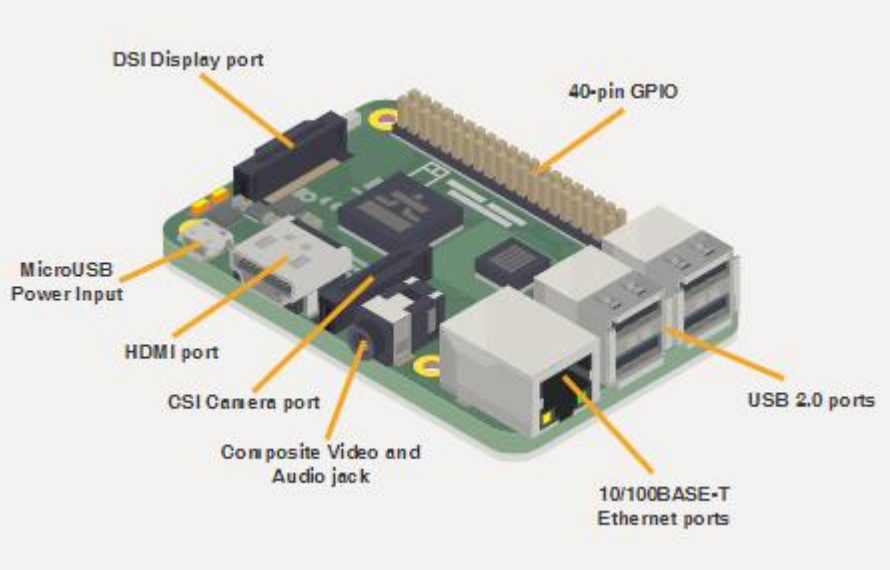


Figure 7: Connections on the Raspberry Pi 3

Source: Adapted from [8]

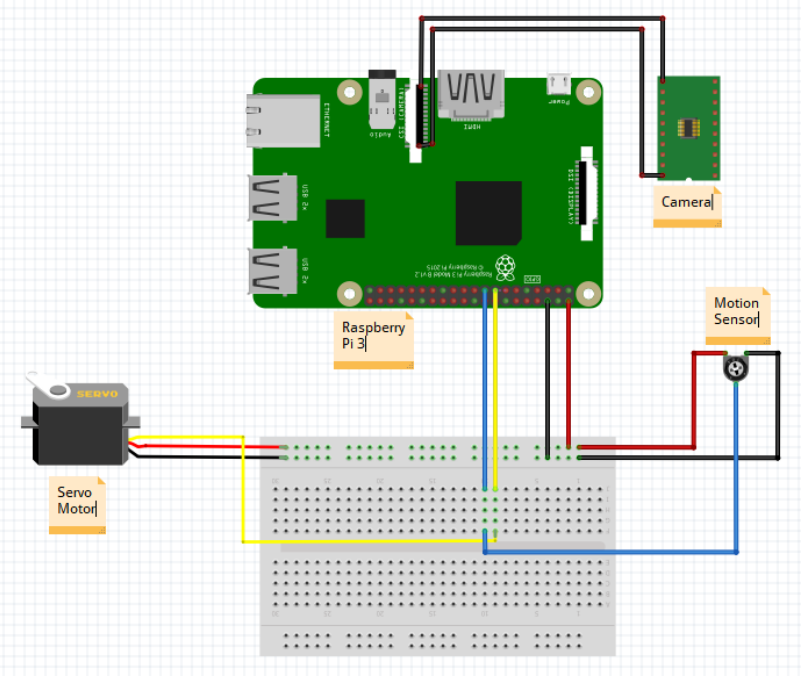


Figure 8: Circuit diagram of the Raspberry Pi with other electronic components

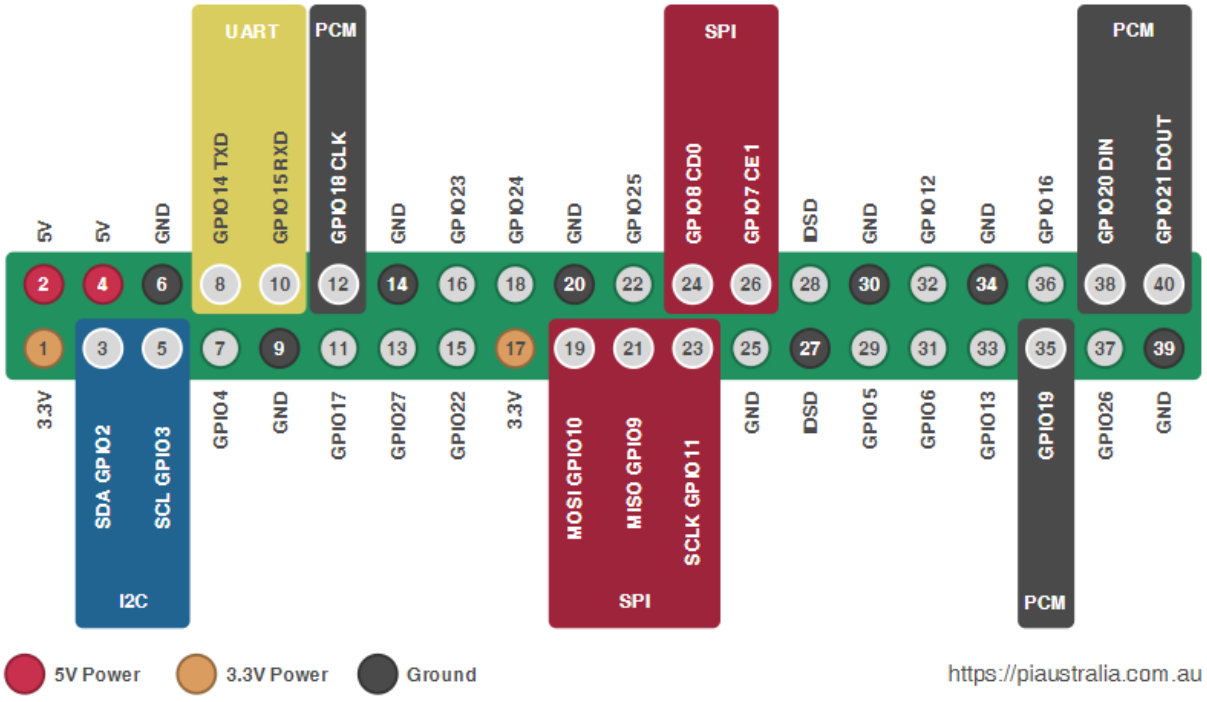


Figure 9: GPIO pins on the Raspberry Pi 3,

Source: Adapted from [8]

5.1.2. Mechanical Setup

As mentioned earlier, a number of components, such as the mount for servo motor, the dispenser case, and the motion sensor cover, were 3D printed using Polylactic Acid (PLA) material. These items were then hot-glued on to the corresponding attachments. Finally, the entire setup was assembled on and attached to a vertical support base as shown in Figure 10.

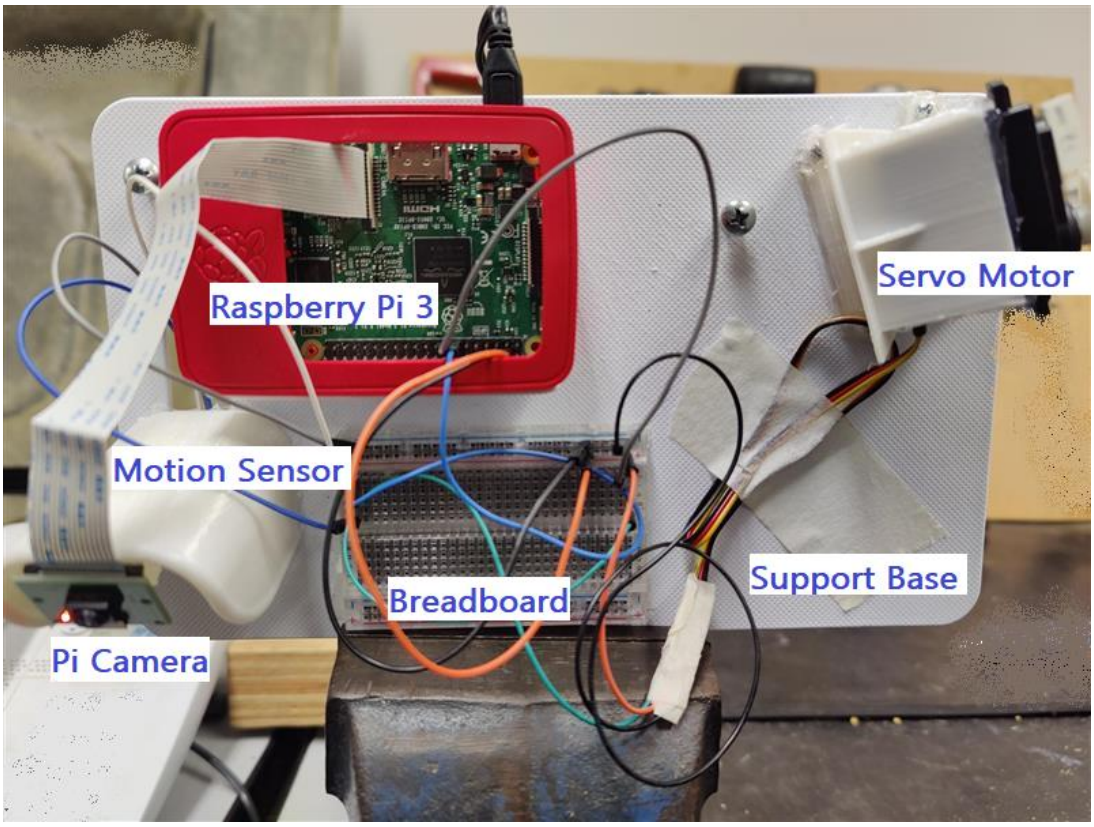


Figure 10. Hardware Setup

5.2. Software Architecture & System Implementation

The software is primarily based on the Raspberry Pi. Listed below are the components of the software:

5.2.1. “main.cpp”

This is the main program written in C++. It handles the feeding algorithm, the motion sensor input and the Blynk connection including syncing of virtual tags V0 to V5, which are described later. It consists of the following functions:

1. “setup”: This function parses the Blynk token ID, starts the Blynk connection to the server, initiates wiringPi for the use of GPIO pins and sets the pin to be used for the Motion Sensor input. The Servo Motor is not configured here as it is used through a separate Python script as and when required.

2. “dispense": This function dispenses feed if there are any feeds left as per the pre-set counter. As mentioned earlier, it calls a python script named “servo.py” to execute the dispensing operation of the servo motor. It then updates the feeds remaining count (V3) and the last dispensed time (V0).
3. “BLYNK_CONNECTED”: This function is automatically executed as soon as a connection to Blynk server is established. The aim is to sync the initial feeds remaining count (V3) to the server and obtain the state of the “Autofeed Enable” (V5) switch from the server.
4. “BLYNK_WRITE(V1)”: This function is automatically executed when the “Manual Feed” (V1) button is tapped on the remote Blynk client. It calls the “dispense” function and syncs back the button state to unpressed.
5. “BLYNK_WRITE(V4)”: This function is automatically executed when the “Shutdown Pi” (V4) button is tapped on the remote Blynk client. It syncs back the button state to unpressed before calling an immediate system shutdown command. The main purpose of having this button is to be able to safely and easily shutdown the Pi when the Pet Feeder needs to be disconnected from power supply, without having to open the desktop view of the Pi.
6. “BLYNK_WRITE(V5)”: This function is automatically executed when there is a change in the “Autofeed Enable” (V5) switch in the server. It then syncs the local variable to enable or disable autofeed.
7. “loop”: This function is meant for endless call and does a few tasks:
    a. It keeps the Blynk interface running and active
    b. It checks for rising edge from the motion sensor. If motion is detected, the motion detected time (V2) is synced with the server and “dispense” function is called if sufficient time has passed since the last feed. This time can be configured but is meant to be lower than the time set for the timed auto-feed functionality.
    c. If the time set for the timed auto-feed functionality has passed since last feed, it calls the “dispense” function
8. “main”: This is the main function which starts when the program is run. It calls the “setup” function and then calls the “loop” function endlessly. From the above, it follows that the “dispense” function can be called in 3 different ways and hence there are 3 modes for operation of the Pet Feeder:
    a. Remote (Manual) feed
    b. Timed (Automatic) feed
    c. Motion triggered (Automatic) feed

5.2.2. “servo.py”

This is the servo motor motion script written in Python and called from time to time by the main program. Standard Raspberry Pi GPIO libraries are used to sets the servo pin as output and initialise a 50Hz PWM signal. The servo starts at the resting position wherein a predefined quantity of feed is already loaded at

the nozzle. It moves the servo into the dispensing position, wherein the nozzle is pointing downwards, and the feed is dropped into a bowl on the floor. It then returns the servo to the resting position after 1 second and the next feed is loaded into the nozzle. At the end of this script, the servo control is stopped, and control returned to the calling main program. This helps in avoiding stray movements of the servo as it keeps trying to adjust its precise position endlessly if the control is not stopped.

5.2.3. Pi Camera setup

The camera setup in this project runs a Real Time Streaming Protocol (RTSP) server which establishes and controls media sessions between devices. (URL). On the Raspbian Stretch simply installing “git” and “cmake” and ensuring they are updated is sufficient for installation. From here downloading and installing a GitHub project created by mpromonet (see URL) utilises an API released for controlling device drivers on Linux systems, more specifically for supporting real time video capturing. This Video4Linux2 or V4L2 for short is used to create this RTSP for streaming the camera device to the Blynk application by simply accessing the mounted location of the device on the Raspberry Pi. This streams the video captures directly to the network available for viewing. For setup, the process is as follows:

1. Open a terminal interface
2. sudo apt update
    - This updates the list of configured sources and downloads package information
3. sudo apt install git cmake
    - This installs Git and cmake packages or ensures the installed packages are up to date
4. git clone https://github.com/mpromonet/v4l2rtspserver.git
    - This clones the directory specified into the present working directory
5. cd v4l2rtspserver
    - This changes into the newly created directory specified
6. cmake . && make && make install
    - This makes and installs the necessary code for the server to run
7. v4l2rtspserver /dev/video0
    - This runs the server application with the designated video device at the directory given
8. vlc rtsp://IP_ADDRESS_OF_RPI:8554/unicast
    - Opens a VLC window of the given server to check if configuration is successful

5.2.4. Blynk Server & Mobile app setup

The Blynk Server on the Raspberry Pi was installed from a GitHub directory and followed a step by step process for installation. This process is as follows:

1. Open a terminal interface
2. sudo apt update

- This updates the list of configured sources and downloads package information

3. sudo apt-get install git-core
    - This installs the git-core package for usage of git
4. git clone https://github.com/blynkkk/blynk-library.git
    - This clones the directory specified into the present working directory
5. cd blynk-library/linux
    - This changes into the newly created directory and sub-directory specified
6. make clean all target=raspberry
    - This sets the program up for the Raspberry Pi environment and cleans any unnecessary files after setup
7. ./build.sh raspberry
    - This runs a setup script that has pre-configured for the Raspberry PI, created by the developers of Blynk.
8. sudo ./blynk --token=INSERT_TOKEN_HERE
    - This runs the Blynk application with the code supplied in the main.cpp and establishes a server to connect to the Blynk Mobile application of the given token.

After the server has been set up, within the blynk-library/linux folder, the main.cpp file exists where the code for the server is based off. Any changes made to the code requires only Step 6 to be run again to compile the code and Step 8 for running the server.

The Blynk Mobile provides an interface for control and monitoring of the IoT device remotely. The app generates a token ID which is needed by the main program in the Raspberry Pi for a successful connection. Usually, the ID is linked to a project and the project access may be shared by multiple users, although editing is only allowed to the creator.

In this project, two modes of data transfer were used:

1. Virtual tags: These tags may store any format of data and can be synced to and from the program running remotely on the Pi. The following tags were used in this project:

    V0: Last Feed time (String)

    V1: Manual Feed button (Boolean)

    V2: Last Motion time (String)

    V3: Feeds Left count (Integer)

    V4: Shutdown Pi button (Boolean)

    V5: Auto-feed Enable button (Boolean)

2. Live Stream: The camera stream set up on the Pi was directly subscribed to by the Blynk app by using the RTSP link. This enabled viewing of the stream of the feeding area on the app.

5.2.5. Raspberry Pi System level setup

In order to be able to use the Pet Feeder in remote situations, the main program has been set up to auto-start in the background on bootup of the Raspberry Pi. As soon as the program is active, it can be controlled from the Blynk mobile interface.

For the purpose of remote access of the Pi, SSH and TeamViewer have also been configured. The only limitation is that on first use in a new premise, the device needs to be connected to the locally available Wi-fi network and hence may require an external display for ease of access.

6. Challenges & Limitations

For the completion of this project, we came across a series of obstacles and challenges we will now outline. These obstacles came in both hardware and software phases. The first challenged we face was from a design perspective. Our original plan was to use an Arduino for the IoT pet feeder, however, the raspberry pi offered easier connection with the internet, in comparison for Arduino, which requires added hardware for internet connection to be achieved. We were also met with a few mechanical and hardware issues, mainly regarding the servo and feed storage component. We originally went with a smaller servo, however, we swiftly realised this would not be big enough to support the storage case. This is largely due to the motor handle not being stable enough when the storage device for the feed is full. Another mechanic limitation is that the storage device is quite small, so you would have to refill the device on a regular basis.

There were also a combination software issues we came across in the creation of the IoT pet feeder. In terms of hardware, our biggest challenge was the transference of information from the Pi back to the end user. Receiving information from the Pi was our biggest obstacle. In our first design, we were planning to use a local private network, however we found it more effective to use the Blynk network for the IoT operations. Although Blynk was good at establishing the network, controlling the device through the phone application was limited. Although we can set up the IoT device to have unlimited features, the testing of these features can be limited without fully purchasing/paying money for the features of Blynk. Currently, we are on a free trial model, that gives us access to a few different options, but these are limited. In addition to this, an IP address needs to be configured for the live stream, where the mobile phone also must be connected to the same network. This limitation can be avoided by using port forwarding on the local router and dynamic DNS.

From a programming perspective, we were writing most of the code in C/C++, which was a challenge considering the group had little to no experience in the programming language (conventions and syntax). We were using a C++ system call to run a python program for servo operations due to time constraints,

as the python program was created before realising a C++ code was to be used in the final project application. We also had difficulties understanding the functions from the libraries, where experimental analysis was used and recorded to validate what functions did.

In terms of our original plan, the final product of our project has changed drastically. Firstly, we switched from an Arduino UNO to a Raspberry Pi, which was a major design switch. The change from Arduino to Raspberry Pi was done to make internet connectivity easier and increase the functionality of the device (for example, attaching a camera for a live feed). We also didn't have an established dashboard with metrics and basic data analysis, this was created after the original proposal. In addition to this, we equip a larger server then first proposed, as weight was a concern for the storage container when filled up. The initial servo was on its limits to holding the desired weight. The servo itself would have been fine, however the way in which the servo's connection point to the pet-feed container was concentrated at a single point which directed the entire pet-feed bottle there causing some cracks after usage. It was decided that a larger servo with a larger contact surface area for weight to be distributed over a larger area, reducing strain from weight, was a more effective option.

Some of these problems that have been outlined were somewhat predicted in our proposal. We had suspected that connectivity would be an issue between the Raspberry Pi and the application being used to set the commands. This was a new area for us, and we made it a focus to put time into this process as it was integral to the effectiveness of the system. We conclude that this was probably the biggest challenge we faced in the project; however, we were able to manage and successfully complete this task. We didn't expect the mechanical problems and had to think of different solutions to combat weight issues we were having. However, there were many parts of the project that went according to plan, including the 3D printing of the servo case and sensor protector, the configuration upon the chopping board to outline our relevant parts and motion sensor successfully deploying according to the code commands.

7. Conclusions and Future Development

The recent technology developments have increased the interaction between humans and smart objects such as Internet of Things. Various researches and advancements have provided IoT developers the proper information to create new developments in which are utilized and implemented in a cost-effective and convenient manner. The latest trends of integrating personal devices and IoT applications becomes a very promising topic in the future. This project created a prototype of an IoT-based pet feeding system that utilizes IoT components such as sensors and cameras that can collect data to the server in real time, thus providing the end user insights and control over the pet feeder via a mobile application. The final product of this project has shown the key advancement of a pet dispensing system in which it uses the latest smart technologies while meeting the growing needs of people having issues with pet obesity and feeding management while away from home. This project's results envision a new approach of operating a pet feeding device while having a small initial cost to set up. Due to some limitations on materials, time, software knowledge, hardware setup and mechanical skills, the current prototype can still further be

improved. For the upcoming developments, this smart pet feeder can be made better by adding more functions and features addressing the limitations mentioned in the section 6 of this paper. Other improvements can also include integrating this system to the smart home system of the pet owner if they have any in such as way will have one central server that can provide insights of data by certain IoT system in one dashboard. Moreover, the smart feeder can also be improved by adding a mechanism to accommodate different types of pet foods in one system and can detect the pet's face though the pi camera and computer vision technology while autonomously decide to feed or not to feed that certain pet.

8. Acknowledgements

In terms of the overall experience of this project journey, each of the team members have done an excellent job of performing their assigned tasks, being open in communicating issues and suggestions as well as supplied commitment to complete the project on time. We also would like to thank Dr. Atif Mansoor and Omar Anwar for guiding us towards the completion of the project.

REFERENCES

[1] K. Patel, "Internet of Things-IOT: Definition, Characteristics, Architecture, Enabling Technologies, Application & Future Challenges", ResearchGate, 2016. [Online]. Available: https://www.researchgate.net/publication/330425585. [Accessed: 27- Sep- 2020].

[2] S. Priyadarshini, A. Bhusan and B. Mishra, "The Role of IoT and Big Data in Modern Technological Arena: A Comprehensive Study", ReasearchGate, 2019. [Online]. Available: https://www.researchgate.net/publication/330023682. [Accessed: 27- Sep- 2020].

[3] "Australia pet food market – growth, trends, industry analysis and forecast (2020-2025)", Mordor Intelligence, 2019. [Online]. Available: https://www.mordorintelligence.com/industry-reports/australia-pet-food-market

[4] C. Dogs, "Dogs Who Are Home Alone All Day | Petfinder", Petfinder, 2000. [Online]. Available: https://www.petfinder.com/dogs/dog-care/dogs-home-alone-all-day/. [Accessed: 27- Sep- 2020].

[5] "The growth of pet obesity," Veterinary Record, vol. 185, no. Suppl 1, pp. 1–3, Nov. 2019, doi: 10.1136/vr.l6498.

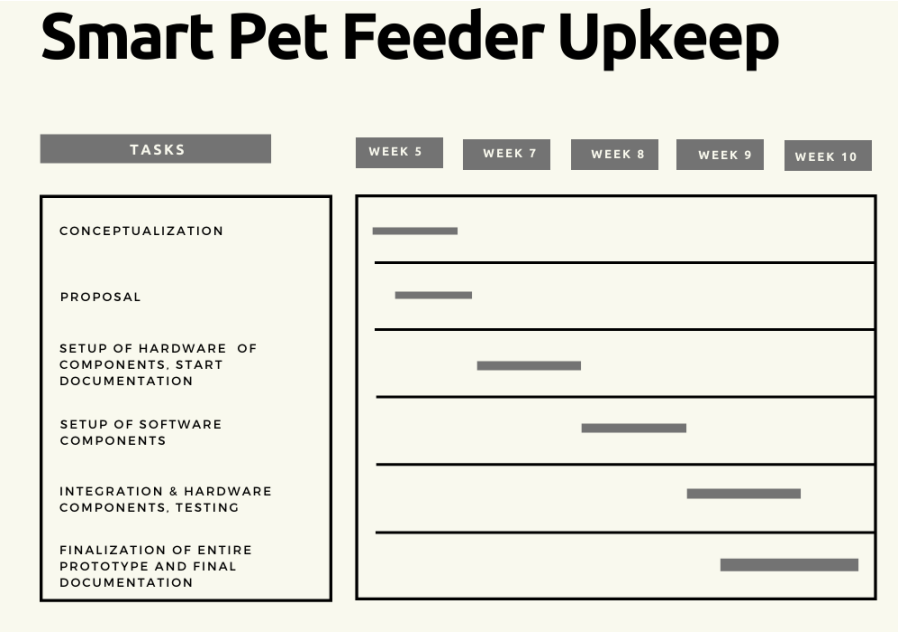
[6] "Documentation for Blynk, the most popular IoT platform for businesses.", Docs.blynk.cc. [Online]. Available: https://docs.blynk.cc/. [Accessed: 28- Sep- 2020].

[7] K. Karyono and H. Nugroho, "Smart dog feeder design using wireless communication, MQTT and Android client", Researchgate.net, 2016. [Online]. Available: https://www.researchgate.net/publication/314105271. [Accessed: 04- Oct- 2020].

[8] "Raspberry Pi 3 Model B", Pi Australia, 2020. [Online]. Available: https://raspberry.piaustralia.com.au/products/raspberry-pi-3-model-b. [Accessed: 11- Oct- 2020].

APPENDIX

I. Project Timeline



II. Distribution of Work

Name of Student	Work Assigned
Jordan Harun	Research and acquisition of parts and software
Clariza Look	Prepare documentation, project management, research of literatures, diagrams
Tom Nicholls	Software, documentation support
Farhad Ahmed	Hardware, circuitry

III. Accountability Documents

Date uploaded at MS Teams/Week Nr: Week 7

Name	**What I contributed to the teamwork this week**
Jordan Harun	• Suggested the change to using Raspberry Pi for Project instead of Arduino • Booted personal Raspberry Pi with Noobs Software (prior to receiving

	one from the University) and explored Python. • Started research into controlling servos with Raspberry Pi • Started research into the possibility of still using circuit.io as the IoT Application
Clariza Look	• Started collecting research and review paper about pet feeders • Organized team meeting • Organized top contents of project report and prepared draft project report
Farhad Ahmed	• Studied the specification sheets and user manuals for the IR Sensor and the Servo Motor. • Researched on the I/O pins of the Raspberry Pi and hardware interfacing with the sensors and the motor. • Started program development for sensor interface, which will be tested during the lab in Week 8.
Tom Nicholls	• Researched into different types of Python code used on similar pet feeder devices • Looked into different code used on Raspberry Pi for a few different functions needed for the project

Date uploaded at MS Teams/Week Nr: Week 8

Name	**What I contributed to the teamwork this week**
Jordan Harun	• Worked on the software code in Python with the team. Researched the use of using gpiozero and RPi.GPIO and the differences before implementing and interfacing with the servo, led and motion sensor • Researched into possible use cases for server and client code for the application and experimented on personal Raspberry Pi.
Clariza Look	• Started writing Introduction of pet feeder final project draft
Farhad Ahmed	• During Thursday lab session, worked with the team to test out the motion sensor and the servo. I worked mostly on the hardware side, interfacing the components with the Raspberry Pi. • Researched on the 3D printing, which the team aims to carry out during Week 9. Undertook training required for printing in Maker's Lab.

Tom Nicholls	• Worked on software code in Python in the team. Helped with the writing, research and experimenting of different codes for different functions. Examples include making the sensor react & the rotator rotate. • Research and developed some 3D sketches for the funnel system to be developed for the Pet Feeder feed holder • Sourced and set up a place to test the pet feeder for when it is completed, and experiment will take place.

Date uploaded at MS Teams/Week Nr: Week 9

Name	**What I contributed to the teamwork this week**
Jordan Harun	• Proposed the use of a third-party interface of Blynk after the suggestion given by Clariza. The choice for using this would alleviate network connection problems of being on different networks. • Set up the working environment for using Blynk on the Raspberry Pi. • Setup a basic LED Circuit for testing the use of the Blynk application. Adapted a C++ code and created a basic interface on the phone application to understand how the application is used. • Experimented with the code to fully understand parameters in the code to later adapt servo and motion sensor. • Brushed up on my understanding of C for code implementation
Clariza Look	• Communicated with team with how the project is going regarding hardware and software updates • Researched for more literatures to be used and continued writing the body of the project
Farhad Ahmed	• Adapted several freely available 3D models for mounts and casings to use in our project with the servo motor and the motion sensor. • Took 3D prints of the mounts and casings and carried out mechanical works to attach all the hardware together. • Wrote a sample program in C to mimic the pet feed release action for the servo motor (to be later ported on to the Raspberry Pi).
Tom Nicholls	• Helped with sample code to make servo motor spin as well as getting the sensor to work through Raspberry Pi. Numerous examples were tried with

	a combination of Python & C code used. • Sourced pellets & drill for construction of model for IoT pet feeder, which will be used in coming week. • Set project timeline and outline for future weeks. Realignment of roles and setting meetings for tuition free week was discussed.

Date uploaded at MS Teams/Week Nr: Week 10

Name	**What I contributed to the teamwork this week**
Jordan Harun	• Worked together with the rest of the team to develop the main program to control the feed through the Raspberry Pi for 3 different situations: Remote (Manual) feed, Timed (Automatic) feed and Sensor based feed with necessary timings and restrictions. • Helped with the Blynk interface setup to pass information to and from the application. • Worked on integrating a local server for experimenting. • Established camera connection and streaming service to Blynk application.
Clariza Look	• Finalized project document report sections including conclusion and parts of appendix • Discussed together with the team regarding the overall project document, things lacking, and things that can be added
Farhad Ahmed	• Worked together with the rest of the team to develop the main program to control the feed through the Raspberry Pi for 3 different situations: Remote (Manual) feed, Timed (Automatic) feed and Sensor based feed with necessary timings and restrictions. • Created the circuit diagram schematics on Fritzing. • Set up program for auto-start in background on bootup of Raspberry Pi • Worked with other team members to attach all the components together onto the board on the back. • Procured new bigger servo, redesigned and 3D printed the mount for it. • Replaced mount and hot glued all printed components for stability. • Added new functionality in the main program – last motion time, remote shutdown and auto-feed enable/disable button.

Tom Nicholls	• Sourced parts for project including nuts, bolts, double-sided tape and chicken feed. • Helped build and attach the components to the chopping board, these components being the raspberry Pi, the sensor, the motor and the camera. • Worked with rest of team to help develop the program to control the feed for the three different phases.

IV. Software Codes

a) MAIN.CPP

```
#define motionfeedTime 60
#define autofeedTime 120
#define BLYNK_PRINT stdout


#ifdef RASPBERRY
#include <BlynkApiWiringPi.h>
#else
#include <BlynkApiLinux.h>
#endif


#include <BlynkSocket.h>
#include <BlynkOptionsParser.h>
#include <wiringPi.h>
#include <iostream>
#include <ctime>
#include <stdlib.h>


static BlynkTransportSocket _blynkTransport;
BlynkSocket Blynk(_blynkTransport);
```

```
#include <BlynkWidgets.h>

using namespace std;

// Global variables to store the system status
bool sensorOldState = 0;
time_t lastmotionTime = 0;
time_t lastfeedTime = 0;
int feedsLeft = 10;
int autofeed = 0;

// Setup function to parse the token ID, start Blynk, initiate wiringPi and set I/O pins
void setup(int argc, char* argv[])
{
static const char* auth, * serv;
static uint16_t port;
parse_options(argc, argv, auth, serv, port);
Blynk.begin(auth, serv, port);
wiringPiSetup();
pinMode(27, INPUT);
}

// Dispense function
void dispense()
{

if (feedsLeft) //Call python script if feeds available & then sync feed time and feeds left
{
```

```
cout << "Dispensing feed.\n";
system("python servo.py");
feedsLeft--;
lastfeedTime = time(0);
char* dt = ctime(&lastfeedTime);
Blynk.virtualWrite(V0, dt);
Blynk.virtualWrite(V3, feedsLeft);
cout << dt << "\n";
}
else cout<<"No feeds left!!\n\n";


}


//Function executed as soon as connection established to server - sync initial feeds & autofeed enable
BLYNK_CONNECTED()
{
Blynk.virtualWrite(V3, feedsLeft);
Blynk.syncVirtual(V5);
}


// Function to receive manual feed command from the Blynk App
BLYNK_WRITE(V1)
{
int pinValue = param.asInt();


if (pinValue == 1) // When pressed, call dispense function & reset button state
{
cout << "Manual Feed.\n";
dispense();
int pinValue = 0;
```

```
Blynk.syncVirtual(V1);
}


}

// Function to receive shutdown command from the Blynk App
BLYNK_WRITE(V4)
{
int pinValue = param.asInt();

if (pinValue == 1) // When pressed, reset button state & then initiate a system shutdown
{
cout << "Remote Shutdown.\n";
int pinValue = 0;
Blynk.syncVirtual(V4);
system("sudo shutdown now");
}


}

// Function to sync autofeed enable command from the Blynk App
BLYNK_WRITE(V5)
{
autofeed = param.asInt();
}

// Endless loop for the embedded application - motion detection & autofeed (timed & motion)
void loop()
{
```

```
Blynk.run(); // Continuously run the Blynk interface
bool sensorCurrentState = digitalRead(27);


if (sensorCurrentState - sensorOldState == 1) // Check for rising edge from motion sensor
{
cout << "Motion Detected.\n";
lastmotionTime = time(0);
char* dt = ctime(&lastmotionTime);
Blynk.virtualWrite(V2, dt); // Sync the last motion time
// Only feed if sufficient time has passed & autofeed is enabled (motion detected case)
if ((time(0) - lastfeedTime) > motionfeedTime && autofeed) dispense();
else cout << "No feed dispensed.\n\n";
}


sensorOldState = sensorCurrentState;


// Only feed if sufficient time has passed, autofeed is enabled & feeds available (no motion detected case)
if ((time(0) - lastfeedTime) > autofeedTime && autofeed && feedsLeft)
{
cout << "Timed Feed.\n";
dispense();
}


}


// Main function to run the setup function and then start the endless loop
int main(int argc, char* argv[])
{
setup(argc, argv);
```

```
while (true)
{
loop();
}

return 0;
}
```

b) Servo.py

```
import RPi.GPIO as GPIO
from gpiozero import MotionSensor, LED
from signal import pause
import time

servoPIN = 22
GPIO.setmode(GPIO.BCM)
GPIO.setup(servoPIN, GPIO.OUT)  # Set servo pin as output

p = GPIO.PWM(servoPIN, 50)      # PWM with 50Hz
p.start(12.5)                   # Initialize at resting position
try:
   p.ChangeDutyCycle(2.5)      # Command servo to move to dispensing position
   time.sleep(1.0)             # Wait for motion to complete
   p.ChangeDutyCycle(12.5)     # Command servo to move to resting position
   time.sleep(1.0)             # Wait for motion to complete
   p.stop()                   # Stop servo control
   GPIO.cleanup()
except KeyboardInterrupt:
  p.stop()
  GPIO.cleanup()
```