[Group 11] Final Project: Smart Dog Collar

Course: CSC 591-791, ECE 592-792 - Internet of Things Architectures, Applications, and Implementation

Submission Date: 4/28/22

1.TEAM MEMBER DETAILS:

1)	Priyam Garg	pgarg6@ncsu.edu
2)	Divyang Doshi	ddoshi2@ncsu.edu
3)	Brendan Driscoll	bhdrisco@ncsu.edu
4)	Jordan Boerger	jwboerge@ncsu.edu
5)	Vishal Veera Reddy	vveerar2@ncsu.edu

Percent Contribution	
Priyam Garg	20%
Divyang Doshi	20%
Brendan Driscoll	20%
Jordan Boerger	20%
Vishal Veera Reddy	20%

Tasks		**Members**				
Topic	**Weight**	**Priyam Garg**	**Divyang Doshi**	**Brendan Driscoll**	**Jordan Boerger**	**Vishal Veera Reddy**
High Level Design	.1	20%	20%	20%	20%	20%
Algorithm Development	.25	20%	20%	20%	20%	20%
Coding	.35	20%	20%	20%	20%	20%
Debugging	.2	20%	20%	20%	20%	20%
Report Writing	.1	20%	20%	20%	20%	20%
Per Student Aggregate Contribution		20 * .1 + 20 * .25 + 20 * .35 + 20 * .2 + .1 + 20 = 20.1	20 * .1 + 20 * .25 + 20 * .35 + 20 * .2 + .1 + 20 = 20.1	20 * .1 + 20 * .25 + 20 * .35 + 20 * .2 + .1 + 20 = 20.1	20 * .1 + 20 * .25 + 20 * .35 + 20 * .2 + .1 + 20 = 20.1	20 * .1 + 20 * .25 + 20 * .35 + 20 * .2 + .1 + 20 = 20.1

We believed that every teammate did their part well and had an equal contribution in the project.

2. INTRODUCTION

Our objective is to create a system allowing the user to keep an eye on their dog / pet and see if they are entering an area, while unsupervised, and interacting with the owner's items. In our specific case, a user has a pet dog which enjoys moving and knocking over a trash can when the user is not present. Our system would allow the user to view the dogs status without having to keep a constant watch over the dog.

3. DESIGN

3.1. OVERALL DESIGN

Our design will feature three end devices: two raspberry pi's to receive data from the MPU sensors on the dog and trash can respectively. Our third device can be a laptop or raspberry pi that will run the MQTT broker, web server for the web page UI, and computation for triangulation. Three BLE beacons are also required to determine the dog's location in the room. Apart from that, we will require one more BLE beacon placed on the trash can to determine if the dog is near the trash can precisely.

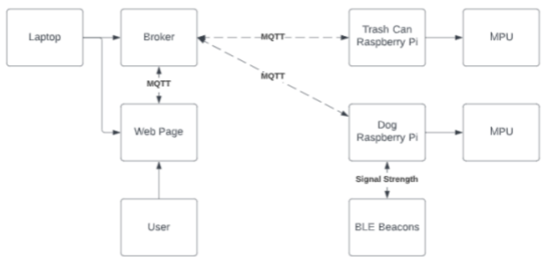

Figure 1. Block diagram of full system.

The user places a raspberry pi with an accompanying MPU sensor on the dog to read the movement data to determine if the dog is moving. This device also functions as a receiver for the BLE beacons. Based on the relative signal strength between the receiver and each beacon, we triangulate the dog's position. The user also places a raspberry pi with an accompanying MPU sensor on the trash can. This will determine the trash can's state (moving or not moving) and if it is in danger of being interacted with. We can assume the trash can is in danger if the dog is near the trash can (determined by the RSSI strength emitted by a beacon on the trash can), and both of them are moving. We then publish this data over MQTT for respective devices to receive and process it.

Now that we have the data for the dog's state, position, the trash can's state, and have determined if the can is being interacted with, we can display that information on a UI. The UI is run on a web server hosted on the MQTT broker device. The web server will be a subscriber to the MQTT topics the other devices are publishing to, allowing it to easily show the relevant information and be replaced if a different UI is preferred in the future. Being a web page also allows it to be accessed from any device on the local network.

4. IMPLEMENTATION

4.1. MQTT

We have used MQTT as a mode for communication between different devices.

4.1.1. RASPBERRY PI DOG

The Raspberry Pi on the Dog acts as a publisher for the topic "ncsu/iot/dogStatus" and "ncsu/iot/InformTrashCanOwner", and "ncsu/iot/DogPosition". Whenever the dog starts moving, detected based on MPU values, it publishes a "DogMoving" on the topic "ncsu/iot/dogStatus". When dog becomes idle again, it publishes "DogIdle" on the topic "ncsu/iot/dogStatus". When the dog is moving, the coordinates of the dog are calculated based on RSSI values from the BLE beacons and published to the topic "ncsu/iot/DogPosition".

4.1.2. RASPBERRY PI TRASHCAN

The raspberry pi on the trashcan acts as subscriber for the topic "ncsu/iot/InformTrashCanOwner". Whenever the dog is nearby it receives a message "DogNearby" from the above topic. If the dog is nearby the trashcan, and the MPU value from the Raspberry Pi on the trashcan starts changing then we know that the dog has started playing with the trashcan. Therefore, this raspberry pi will then publish a message "DogStartedPlayingWithTrashcan" on the topic "ncsu/iot/TrashCanInDanger"

4.1.3. LAPTOP

The laptop acts as a subscriber for all the topics that is ["ncsu/iot/DogWalkingStatus", "ncsu/iot/TrashCanInDanger", "ncsu/iot/DogCoordinates", "ncsu/iot/InformTrashcanOwner"]. It prints the messages to show the working of the system.

4.1.4. UI

The UI acts as subscriber for "ncsu/iot/DogWalkingStatus", "ncsu/iot/DogCoordinates" and "ncsu/iot/TrashCanInDanger", so that user can see on the webpage when the status of the dog at various time as well as track the dogs location.

4.2. Trashcan Setup

The trashcan had the simplest implementation of any portion of this project. It used a Raspberry Pi 3B with a MPU6050 to collect accelerometer data. The wiring diagram for this pi can be seen in figure __.

The code uses two helper functions. The first is a calibrate() function, which takes the accelerometer readings at rest and adjusts them to zero. This is useful as it removes acceleration due to gravity from any calculations. The second helper function was a getMag() function which gets the magnitude of the acceleration across the x, y, and z axis. In the main loop of the trashcan pi, it takes a moving average of the getMag() readings and if the readings are above a certain threshold it will publish the message "DogPlayingWithTrashcan" to the broker. Afterwards, if the accelerometer readings go back down below the threshold it will publish the message "TrashCanSafe".

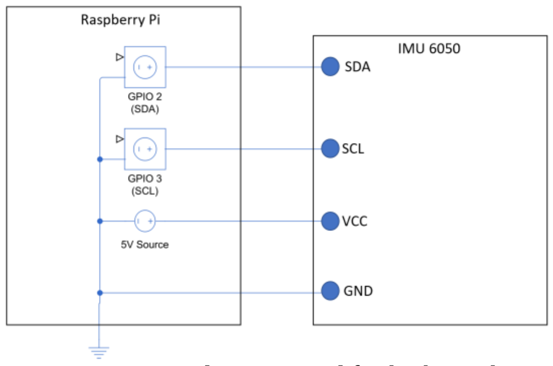

Figure 2. Wiring diagram used for both raspberry Pi's used in this project.

The trash can also has a BLE beacon (Android smartphone in our case) placed on top of it. If the RSSI value received by the Raspberry Pi on the dog is less than -61 (determined by experiment) we publish a "DogNearby" message on MQTT topic "ncsu/iot/InformTrashCanOwner"

4.3. Bluetooth Beacons

We have set up 4 bluetooth beacons using our smartphones for this experiment. Three beacons are placed at various locations in the house for calculating the approximate coordinates of the dog using triangulation. The code runs on the Raspberry Pi that is placed on the dog. We are using the Bluez library to scan for nearby beacons using the namespace. Based on the RSSI values received by these beacons, we calculate the coordinates using a triangulation equation as mentioned in section 4.4.2.

We have one more bluetooth beacon setup on the trash can so that whenever the dog is nearby the trash can, it can alert the owner about this event. The code runs on the Raspberry Pi placed on the dog and it publishes the message when the dog goes nearby the trash can determined by RSSI value. In our experiment, when the RSSI value goes above -61, we say that the dog is near the trash can. We have also employed a technique of waiting till 3 consecutive RSSI values greater than -61 to determine if the dog is nearby to eliminate the variations observed in the RSSI values.

4.4. Dog Setup

4.4.1. MPU

The raspberry pi used for the dog had the same setup and wiring diagram used for the trash can pi as seen in figure__.

Detecting when the dog was moving or resting was more tricky than setting up the trashcan's code. We couldn't just take the total acceleration magnitude and set a threshold, because the dog will always be experiencing acceleration due to gravity and unlike the trashcan where we can factor out gravity, the dog can always roll over onto its side and this would undue and offsets from a calibration function like what was used in the trash can. To solve this we worked with gravity instead of trying to factor it out, and instead of just reading the acceleration magnitude we checked the difference between the acceleration magnitude and the acceleration due to gravity. If the acceleration was ever significantly greater than or less than what would be expected of gravity, the dog's status was changed to moving.

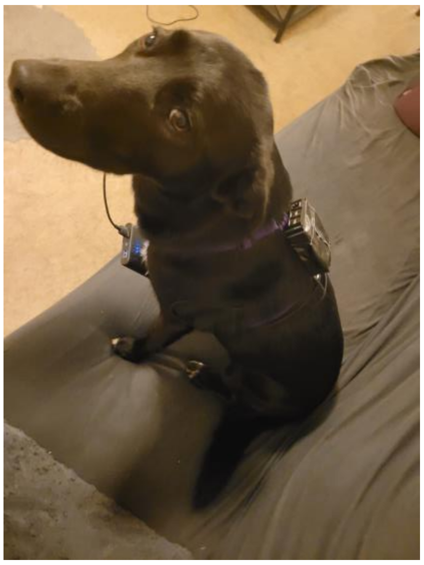

Figure 3. Dog wearing the smart harness. Raspberry Pi on back, battery pack on front.

Next, in order to prevent premature setting the dog's status back to idle, a 2 second timer was set whenever the dog moved. This timer would only count down if the dog remained still, and the dog moving again would reset the timer. The dog's status only returns to idle once the timer reaches 0. This means that once the dog begins moving it's status will only return to idle if it remains still for a few seconds, this is to prevent the dog's status from alternating between moving and idle whenever the acceleration magnitude reenters the threshold region.

4.4.2. RSSI TRIANGULATION

To calculate the position of the dog moving in a room, we have used RSSI Triangulation. We had set up 3 BLE beacons (Android smartphones) in known positions determined according to the apartment map. The Raspberry Pi on the dog was responsible for scanning and reading the RSSI values emitted by these beacons, and calculate an approximate position for the dog based on the following equations.

Let the positions of the BLE beacons be:-

$$B1 = (x1, y1)$$
$$B2 = (x2, y2)$$
$$B3 = (x3, y3)$$

Let the coordinates for the point to be identified (position of dog) = (x,y)

Using the RSSI values emitted from the BLE beacons we can find out the approximate distance between the BLE beacon and the receiver (Raspberry Pi on the dog) by using the following equation :-

$$Distance = 10\text{^}((Measured\_Power - RSSI)\ /(10 * N))$$

We have taken N = 4 for our experimental setting. Also, the Measured Power is a factory-calibrated, read-only constant which indicates what's the expected RSSI at a distance of 1 meter to the beacon.

Let r1, r2, r3 be the approximate distance from B1, B2, B3 respectively calculated using the above equation
Using the equation of circle we get:-

$$(x - x1)^2 + (y - y1)^2 = r1^2$$
$$(x - x2)^2 + (y - y2)^2 = r2^2$$
$$(x - x3)^2 + (y - y3)^2 = r3^2$$

On subtracting equation 1 from 2 we get:-

$$x(2x1 - 2x2) + y(2y1 - 2y2) = x2^2 - x1^2 + y2^2 - y1^2 + r1^2 - r2^2$$

On subtracting equation 2 from 3 we get:-

$$x(2x2 - 2x3) + y(2y2 - 2y3) = x3^2 - x2^2 + y3^2 - y2^2 + r2^2 - r3^2$$

In the above two equations, we know the RHS as we have all the values available with us, and we know the coefficients of x and y also. By solving these linear equations at runtime, we can determine the current coordinates of the dog represented by (x, y)

4.4.3. DATA RECORDING

In order to collect data to help visualize the dog's different movements, a separate python script was written. When run, the script prompts the user to select a movement type. The options given are Sit, Laydown, Spin, or Walk. These are the different types of movement the dog has been trained to perform, and the user will input the option that they will command the dog to do. Next the user will enter a filename that the data will be saved to. Once the file name is given, the program will automatically start collecting data from both the accelerometer and gyro, and will continue collecting data until the dog remains still for 2 seconds. The user should try to keep the dog standing still until ready to record, then have the dog stay in place again after the trick is performed until the program stops running. The code to detect the dog's status of moving or idle was the same used in the previously discussed MPU code.

4.5. USER INTERFACE

The user interface is designed to be easily stood up, taken down, and accessed. A web page was chosen to display user information. The web page is hosted on the third device in this project which is hosting the MQTT broker. This allows it to easily access the topics. The web server is hosted using nodejs on port 8080 of the device. In order to keep the interface between the end devices and the UI separate, the web page subscribes to the MQTT topics the dog and trash can devices are publishing to. This allows the UI to be abstracted away from the physical system.

The webpage is written in html and javascript. Html provides the general structure of the web page while javascript allows us to access the MQTT topics and draw our dogs path on the site. As data is received for each topic, it is updated on the UI. The main component of the UI is a large map which displays the path of the dog as it moves through the room. The points of this path are determined from the BLE signal strengths giving us an x and y component. These points are also listed on the UI for a record of travel. Status for the trash can and dog states are also displayed directly on the page for easy viewing.

5. Results & Discussion

5.1. Dog MPU Recordings

Looking at figures _ - _ the MPU readings can be seen from when the dog performed different tricks. While creating a neural network for determining the dog's movement was outside the scope of this project, we can see when the data is visualized that the different tricks to have their own distinct curves and patterns. One can easily see how a neural network may be able to differentiate between the different tasks, and it's possible that with good feature extraction algorithms that even a simple machine learning algorithm such as an SVM may be able to differentiate between the different classes.

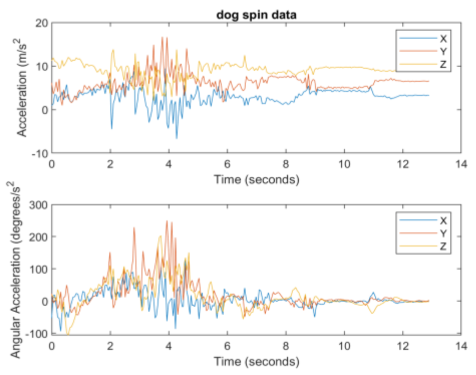

Figure 4. Accelerometer & Gyro data recorded from the dog performing different tricks, Spinning in a circle.

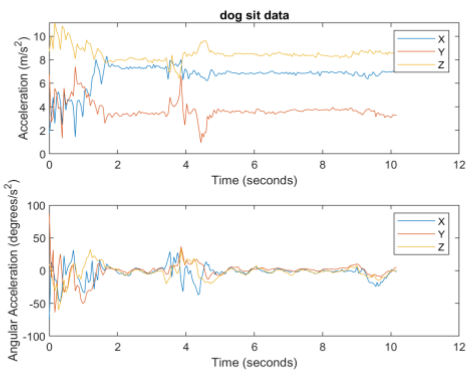

Figure 5. Accelerometer & Gyro data recorded from the dog performing different tricks, Sitting down.

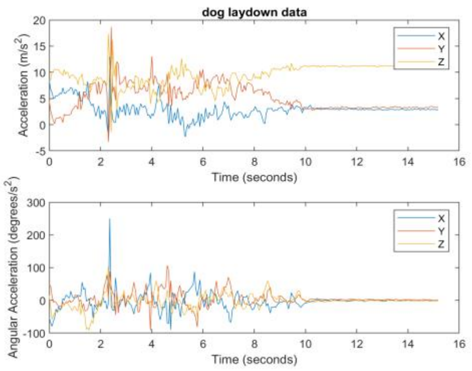

Figure 6. Accelerometer & Gyro data recorded from the dog performing different tricks, Laying down.

5.2. Dog + Trashcan Interaction

When the dog reaches near the trash can the RSSI value from the beacon present on top of trash can has a small value (-61 in our case). If this is the case, we determine that dog is near the trash can and a "DogNearby" message is publish on MQTT broker, which is displayed on the Laptop subscriber.

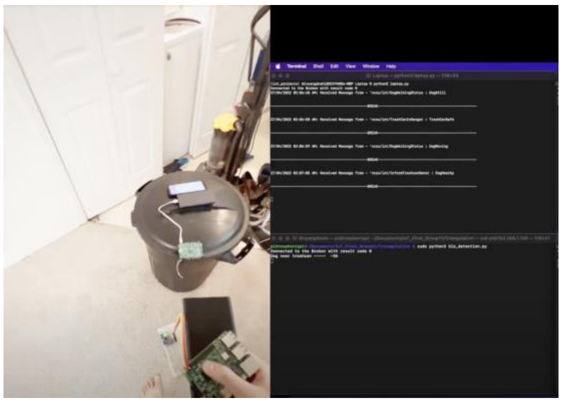

Figure 7. Dog Nearby Trashcan

When the dog is near the trash can and it starts playing with it (determined by the values from MPU), a message “DogPlayingWithTrashCan” is published on the MQTT broker, which is displayed on the Laptop subscriber.

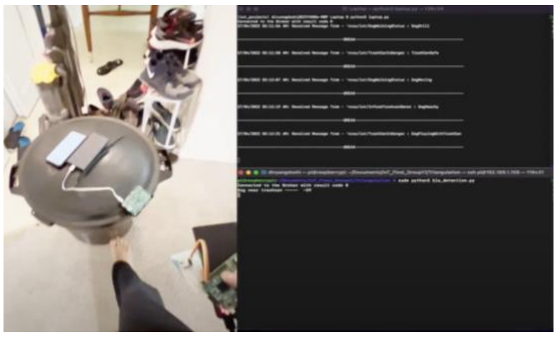
Figure 8. Dog Bumped the Trashcan

5.3. RSSI Position Tracking

When the dog starts moving around the house, we get the approximate position calculated from the BLE Beacons which are plotted on the web page showing the points visited by the dog in the house.

6. Links

Video Link:
Github Link:

7. Related Work and References

[1] https://pimylifeup.com/raspberry-pi-bluetooth/#:~:text=%20Using%20Bluetooth%20within%20the%20Terminal%20%201,to%20scan%20for%20other%20devices..%20By...%20More%20

[2] https://dev.to/ivanmoreno/how-to-connect-raspberry-pi-with-hc-05-bluetooth-module-arduino-programm-3h7a#:~:text=1%20Basic%20connection%20between%20arduino%20and%20raspberry%20pi,to%20leave%20the%20application.%20Skipping%20tty%20reset%20

[3] https://www.lukinotes.com/2018/10/raspberry-pi-bluetooth-connection.html

[4] https://everything2.com/title/Triangulate

[5] https://iotandelectronics.wordpress.com/2016/10/07/how-to-calculate-distance-from-the-rssi-value-of-the-ble-beacon/