University of Toronto

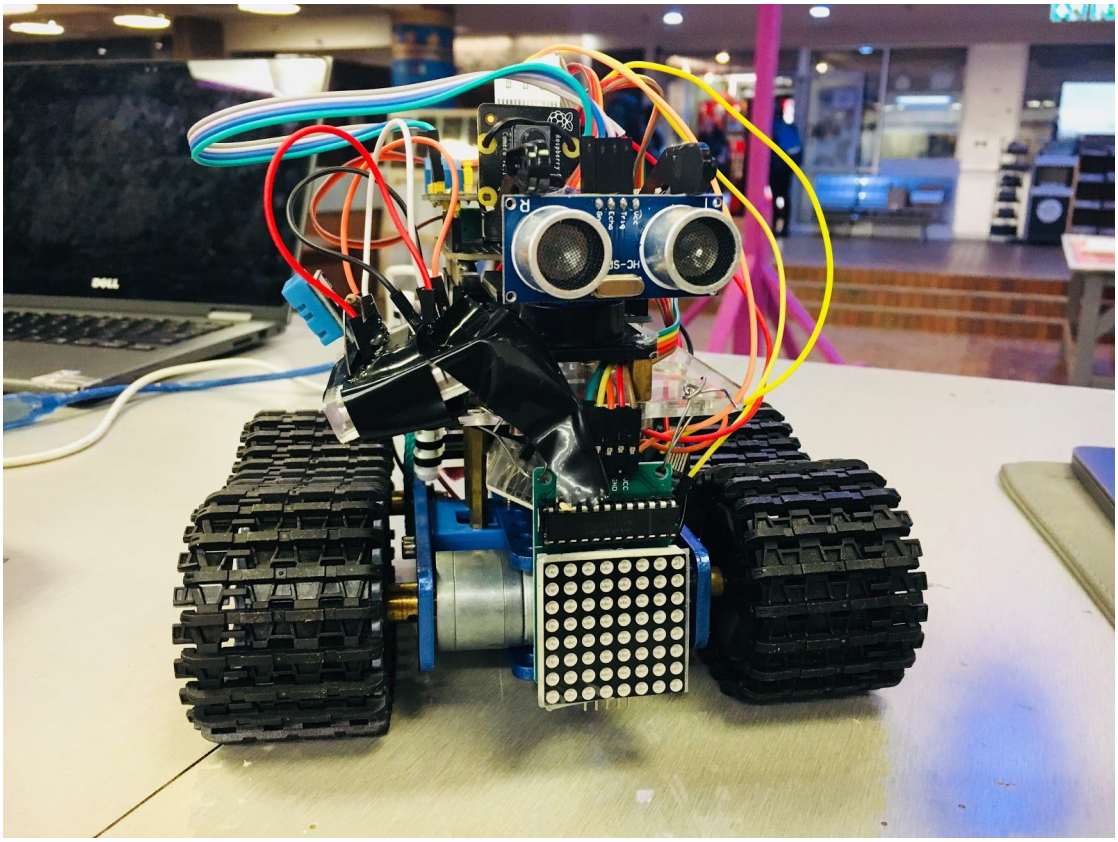
The Pet Rock Robot

prepared by

Team 17
Logan Rooks (1002353071)
Naireen Hussain (1002169422)
Saeed Chehabeddine (1002196036)
Najah Hassan (1002204893)
Yingxue Wang (1002177477)

prepared for

Instructor: Omid Youssefi



MIE438 Project Final Report
April 11th, 2018

Table of Contents

1.Summary of Project Proposal
    1.1 Problem Statement
    1.2 List of Proposed Functionalities
    1.3 List of Required Parts
2. Modification from Project Proposal
    2.1 Modified Functionality
    2.2 Updated List of Parts
3. Choice and Usage of Microcontrollers
    3.1 Raspberry Pi
    3.2 Arduino Uno
    3.3 Arduino Nano
4. Sensors and Hardware
    4.1 Temperature Sensor
    4.2 Ultrasonic Sensor
    4.3 Camera Module
    4.4 Speaker and SD Card Module
    4.5 DC Motors
5. Structure of Code
    5.1 Input from the User
    5.2 Hierarchical Structure of Code
    5.3 Information Stored and Calculated on Pi
    5.4 Information Stored and Calculated on Arduino
    5.5 Information Stored and Calculated on Nano
6. Additional Improvements
7. Evaluation of Performance
8. Conclusion

1. Summary of Project Proposal

1.1 Problem Statement

To address the issue of a lack of autonomous toys that exist in the market today, our team proposed to design a companion robot for children that could be treated as a pet. A device like this would aim to mimic the comfort and companionship that a pet could provide a child without the added responsibility of looking after a living animal.

1.2 List of Final Functionalities

- Object detection and tracking
- Object following
- Temperature sensing
- User identification
- Speech reaction to pre-programmed situations
- Emotion recognition

1.3 List of Required Parts

- 1 Raspberry Pi 3
- 1 Arduino Uno
- 1 Arduino Nano
- 2 DC motors
- 4 Wheels
- 1 Raspberry Pi Camera Module
- DTH Temperature and Humidity Sensor
- 1 Speaker
- Ultrasonic sensor
- LED Dot Matrix Display
- External Server

2. Modification from Project Proposal

2.1 Modified Functionality

Six functionalities were proposed in the original proposal, with a seventh additional functionality being added. Some of the functionalities were required, while others were optional and were organized as such:

Required Functionalities:

1. Path Finding and Mobility
2. Temperature and Humidity Sensing
3. Facial Recognition
4. Ability to Speak

Additional Functionalities

5. Mood Detection
6. Face Display

7. Sleep Mode

All functionalities, both required and additional were accomplished.

2.2 Updated list of Parts

2.2.1 Change in Microcontrollers

The most computationally expensive aspect among our functionalities would be computer vision. Due to this reason, we chose Raspberry Pi 3 as our microcontroller due to its recent advances in net architectures and open-source community supports.

However, in our process of implementation, we notice that the Raspberry Pi(RPi) itself may not have enough computational power to process both computer vision tasks and robot navigation. Therefore, we decided to use an Arduino UNO microcontroller to perform the navigation task using other sensors and actuators that were involved in this project. Arduino UNO is an inexpensive, cross-platform microcontroller that has its own IDE that can be used on various operating systems. This allowed our team to contribute to the algorithm planning and implementation. Furthermore, serial communication between Arduinos and Raspberry Pi is possible. This meant that we could have both microcontrollers working together. To control the various sensors and actuators that we needed, the available pins on the Arduino UNO were not sufficient. Thus, we had to use an additional Arduino Nano microcontroller to control the temperature sensor. Arduino Nano is inexpensive, and was readily available because our team owned one. We used I2C communication between the Arduino UNO and Nano.

2.2.2 Use of SD Card for Speaker Module

The original plan was to load audio files to the arduino memory for playback. This, however, proved to be incredibly difficult. Thus, the SD Card module and SD card were introduced in order to achieve the required functionality. The audio files were stored on the SD card and accessed by the microcontroller when needed.

2.2.3 Serial Communication between Raspberry Pi and Arduino

In our initial design, the use of an Arduino had not been anticipated. When we decided to use both the Pi and the Arduino Uno, a method of communication between the two had to be established. A serial communication method was used because there was a lot of information that we wanted to send across the microcontrollers. Using parallel communication would require too many pins and it seemed more space and time efficient to use a serial communication.

2.2.4 I2C Communication between Arduino Uno And Nano

We later on decided to use an Arduino Nano in addition to the RPi and Uno microcontrollers. We chose to implement I2C communication between Arduino Uno and Nano, as both of them are based on the ATmega328P processor, and can be programed in the same environment. We used Pin 4 and 5 on the Uno board(master) to control the Nano through analog Pin 4 and 5, acting as SDA and SCL signals. The two boards were also commonly grounded, and the Nano was powered by the Uno board. I2C communication is low cost, and only takes up two analog pins. Although it has the disadvantage of being slow in data transmission, it performs well enough to serve this project's purpose.

3. Choice and Usage of Microcontrollers

In this project, we combined 3 microcontrollers, a Raspberry Pi, an Arduino UNO and an Arduino Nano, to achieve our goal. The functionality description and pin assignments are given in the following sections.

3.1 Raspberry Pi

The most computationally expensive aspect among our functionalities would be computer vision. Due to this reason, we chose Raspberry Pi 3 as our microcontroller due to its recent advances in net architectures and open-source community support.

Main Tasks:

1. Performs face recognition of the owner.
2. Obtain target position and distance from the robot.
3. Serial communication with Arduino UNO.

Pin assignment:

Since all the sensors are connected to the Arduinos, none of the GPIO pins were used. However, we did connect the RPi camera module to the SCI Camera pins, in order to detect motion in a live stream. Moreover, the Arduino Uno was connected to the RPi via usb to allow for serial communication.

3.2 Arduino Uno

Arduino UNO is an inexpensive, cross-platform microcontroller. Serial communication between the Arduino UNO and Raspberry Pi is achievable so that we could have both microcontrollers working together.

Main Tasks:

1. Control and read from Ultrasonic sensor to avoid colliding into obstacles.
2. Control motor driver for wheel-motion.
3. Facial expression display using LED Matrix.
4. Serial communication with Raspberry Pi.
5. I2C communication with Arduino Nano.

Pin assignment:

DC Motors	Enable Bit	Voltage bit
Motor 1	Pin 3	Pin 12
Motor 2	Pin 11	Pin 13
Ultrasonic Sensor	Trig	Pin A0
	Echo	Pin A1

Temperature Sensor	A3	
LED Matrix Display	DIN	Pin 0
	CLK	Pin 1
	CS	Pin 2
Vibration Motor	Pin 4	
I2C Communication	A4	A5

3.3 Arduino Nano

Main Tasks:

1. Control and read data from the temperature and humidity sensor.
2. Read from the SD card for audio files and play the sound through speaker.
3. I2C communication sensor with Arduino UNO.

Pin assignment:

SD Card Reader	Pin 4	CS
	Pin 11	MOSI
	Pin 12	MISO
	Pin 13	SCK
Speaker	Pin 9	Signal
I2C Communication	A4	A5

4. Sensors and Hardware

This section introduces the sensors used in our design, and explains the role that they played in this project in detail.

4.1 Temperature Sensor

Temperature and humidity sensing was done with a DTH sensor. Since these are not rapidly changing quantities, no filter was applied to the stream of incoming values. The analog pins A3 was assigned to the sensor to obtain environmental data. The sensor constantly checks the the environment temperature.

4.2 Ultrasonic Sensor

Controlled by two analog pins, A0 and A1 by Arduino Uno, the HC-SR04 ultrasonic sensor was used to detect the distance at the front to prevent the robot from running into objects. In our program, the relative position and distance information of the target were collected through the camera module controlled by RPi, as it is more reliable and accurate compared to using the ultrasonic sensor. We decided to constrain

the effect of the ultrasonic sensor to within 30cm. That is, if the reading of ultrasonic sensor is less than 30cm, which means there is either an obstacle, or the target at the front of the robot, the robot will stop forward motion and start to execute obstacle avoidance algorithm.

4.3 Camera Module

The camera module is connected to the Raspberry Pi and is used for both facial recognition and mood detection, as well as object following.

4.4 Speaker and SD Card Module

The speaker and SD Card module are used in order to give the user audio feedback. The SD Card module has a set of preloaded audio files. The audio files were in a WAVE format and were read using the TMRpcm library.

4.5 DC Motors

Two DC motors were controlled by Arduino Motor Shield Rev3 through pins 3, 11, 12, 13 on the Arduino Uno board. To control the direction and speed of the motions, we used PWM on both motors, with a duty cycle within range (0, 255). The motor shield was directly driven by the Arduino microcontroller.

5. Structure of Code

Main operating flow

a. Target detection and following, including obstacle avoidance
b. Facial recognition, with verification of face with audio output
c. Cold temperature detection, with shiver and audio output
d. Nap if nothing is required

The initial step is to first determine if there is even anything to be processed. To save computational power, we only begin processing if there is motion detected. Once motion has been determined, we check if the predetermined object present. This is done by determining the largest contour of the predetermined object, that can be drawn over a green section of the image. Based on the size and location of the contour, image offset and distance to the object is calculated. Offset of the tracker is calculated relative to the center of the image. This will provide information on whether the robot needs to turn left or right. Since we want the object to appear in the center of the image taken by the Raspberry Pi, if the contour of the object is off center, the robot will correct its trajectory accordingly. The radius of the contour will provide an approximate estimate as to how far it is. If the size is too small, the robot will go forward until a desired distance is achieved.

Though this rudimentary approach does work, there are some bugs. For one, we assume that the object is the largest contour in the field of view. If there is another object of similar contour closer to the Rpi then the Rpi will begin to track that instead. Moreover, the distance calculation assumes that the object is always seen face on. If the object is angled above the direct line of sight of the camera, it will be slightly inaccurate.

5.1 Input from the User

The input from the user are a set of images that can be used for user identification. This helps the algorithm figure out which one of the people in the room is it's user. The user can also input an object to follow. Thus, when the robot detects the object in its field of view, it can follow it.

The other inputs to the system are based on the robot's sensing of its environment. This includes temperature readings and using the ultrasound sensor to determine whether or not an object is blocking it's view.

5.2 Hierarchical Structure of Code

The operation flow of the robot is shown in the diagrams below, separated into RPi and Arduinos. The RPi acts as the master that sends instructions to the Arduinos and determine actions that the robot takes.

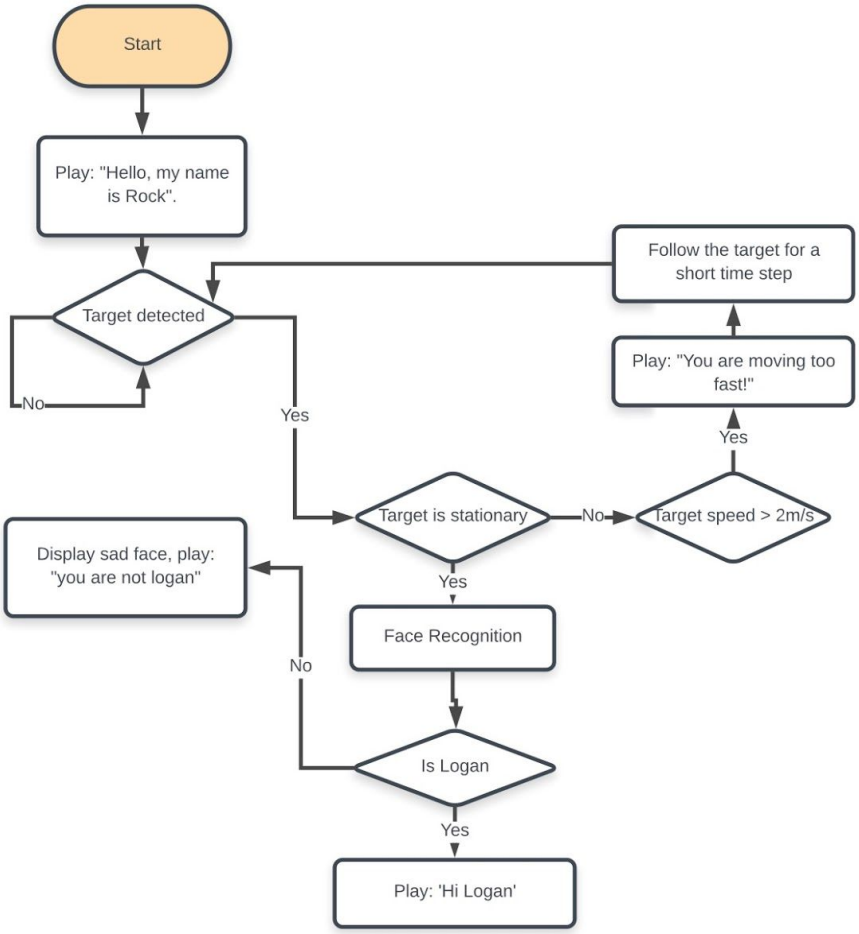


Figure 1. Flow Diagram of RPi control

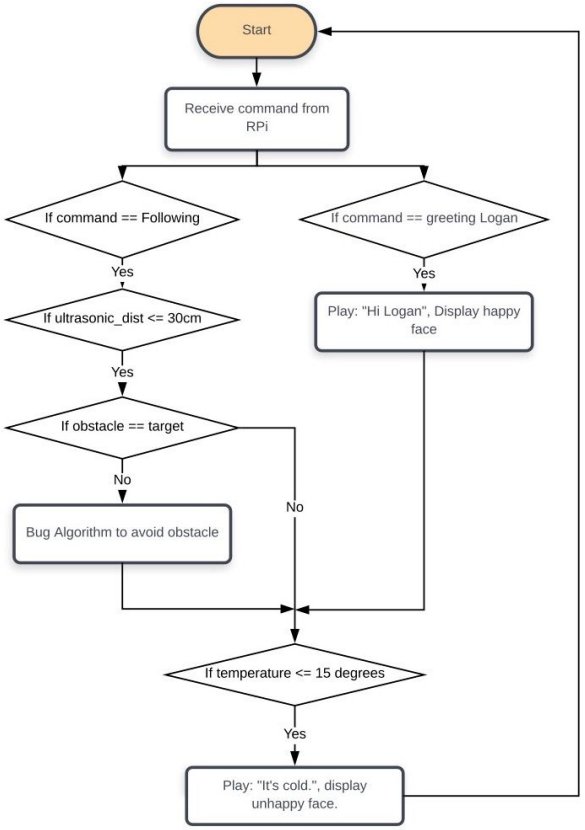


Figure 2. Flow Diagram of Arduino Uno and Nano control

5.3 Information Stored and Calculated on Pi

The RPi was predominantly used for handling computer vision. This had two sub tasks, object detection, and facial recognition. However, a hierarchical approach was used to save computation.

The computer vision begins with a simple motion detection algorithm. For this portion, a grayscale copy of the frames is created and Gaussian blur applied. The difference between the current frame and a moving exponentially weighted average of past frames (moving to adjust for gradual lighting changes) is thresholded (by a parameter experimentally determined) to become a binary image of integers 0s and 1s, where 1s indicate a significant change in color (i.e. movement). An algorithm then looks for the largest patch of 1s and if the bounding contour's area is larger than a predefined minimum area (we only care about significant changes, which occur when people move in view), then the robot determines motion is detected. It is only until then that the other more expensive algorithms for object and face detection are applied, since now we have something potentially interesting in the scene.

The following object detection algorithm is used to detect and localize a single round black object relative to the robot. For each of the frames coming from the camera module, the JPG image would be converted to HSV, due to its robustness to lightning changes. Then, a mask would be applied to identify circular regions of a specified color. In our case, we chose to track black objects. We used a simple detection method,

assuming that the largest black circular contour was the object that we were trying to track. To prevent the Pi from tracking other black objects that would enter its field of view, we had a simple check to ensure that the contour of the last image would overlap with the contour in the next image. This makes an assumption that the object was moving slow enough that their contours would overlap.

Once the object was detected, its offset from the camera's center could be calculated, as well as its approximate distance from the robot. Its distance was calculated using a reference image, previously taken, where the distance of the object to the camera was known. Since the width of the tracking object is also known, its possible to determine the object's perceived pixel width. As this varies as the object moves, its possible to use ratios to then determine the object's new distance. However, error is introduced since this method assumes that the object is always seen face on. This is why the robot always moves to keep the tracked object in the center of its field of view, as that is when the distance information is the most accurate. The distance, x - offset and y - offset is then sent to the Arduino Uno for handling motor control.

Face detection is implemented in its own hierarchy. It is only applied once motion has been detected and then ceases, meaning the target is probably stationary. The frames captured are converted to grayscale and a simple cascading classifier looking for facial features is applied to the image. While this algorithm can detect multiple faces, to simplify the workload, we decided to only fully process one face per frame and thus it is assumed that the largest (also closest) face is of most interest. The cascade classifier returns a bounding box around the perceived face plus a threshold increasing or decreasing its size. The frame is then cropped to the region of interest and sent along the vision pipeline.

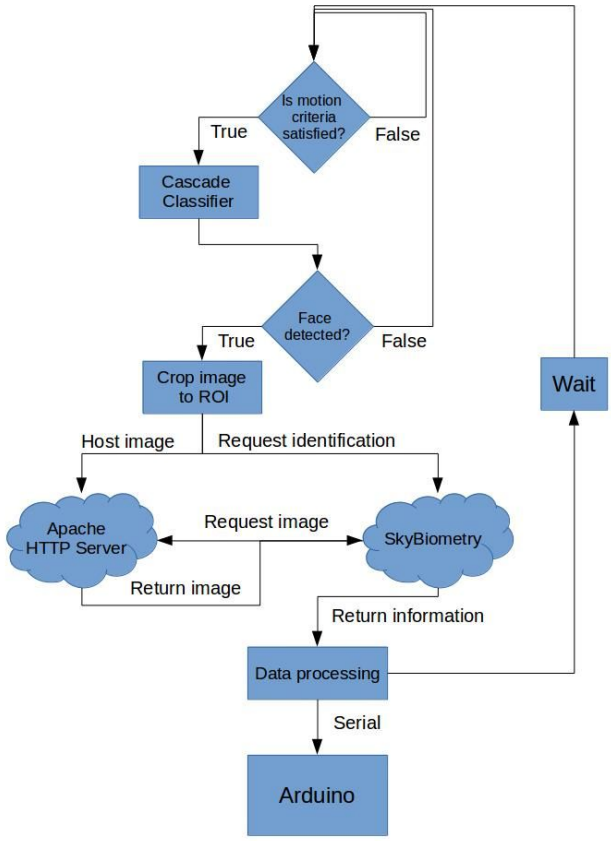


Figure 3. The conditional hierarchical flow of face detection and identification on the Raspberry Pi.

In the proposal, we believed further on-board processing of the face image would be necessary before sending it off to the server. We proposed this would be done with a small convolutional neural net, that was pruned and used quantized integer weights to increase the speed of inference. While the network we trained was fast, it did not add much since the cascade classifier performed better than expected. It was therefore not necessary to add the neural net to the on-board pipeline.

As we stated in the proposal, we used the face detection client SkyBiometry due to the free 5000 calls/month to their servers and their Python API. To recognize faces in an image with the API, the image must be hosted on a server and the URL provided. To do this, we transformed the Pi into a small web server which allowed the SkyBiometry servers to retrieve the images that were hosted on our Pi. This was done by saving a single captured frame of interest to a specific

folder. Each call took about 1 to 2 seconds to receive the returned information. Since we only had a limited number of calls, only one call was made after the criteria from the motion detection was met, before waiting a specific amount of time (usually about 5 seconds) before allowing more calls. Not only does SkyBiometry detect faces, but also attributes such as the person's gender, mood, location of facial features, and whether or not they are wearing glasses, among many others. However, since we were only interested in the identity of the individual detected, we processed and simplified the data into a format able to be sent to the Arduino. Also, along with the predictions for each attribute, SkyBiometry also provides a confidence level for each prediction and an estimated threshold on the confidence based on the quality of the image. This was used to create a binary integer value for each desired attribute that was sent along with the data (specifically, right before) to inform the Arduino whether or not it should trust the proceeding value. In the context of face identification, if the confidence was too low on the proposed identity (selected by choosing the identity with the highest confidence), the integer zero would be sent before the predictions and the Arduino would know that it is currently looking at someone it hasn't seen before. An individual becomes "known" once the SkyBiometry network has trained on several images of the person's face. We used about 15 pictures of Logan in various lighting settings, with and without glasses, to have the robot robustly able to identify him from others.

5.4 Information Stored and Calculated on Arduino Uno

The Arduino Uno was responsible for all the processing that required the use of the sensors and all the outputs like the motors and LED display. The speaker was connected to the Arduino Nano and will be discussed in the next section.

Through serial communication, the Arduino obtained five items of information from the Pi. These are listed as follows:

1. Who the user is
2. Whether or not a face has been detected
3. Whether or not a command to follow the user has been sent
4. The x and y offset of the target from the center of the robot's field of view (i.e. Angle it needs to turn to reach target)
5. Distance that needs to be moved to get to target

If the follow command had been given, the robot would call the *'follow'* function. This was combined with an obstacle avoidance function that was based on a bug algorithm. This meant that if the ultrasonic sensor detected anything in its path ahead that was less than 30 centimeters away, it would begin to turn and head towards the target. It would then continue moving forward based on the distance it would get from the RPi. The function would calculate angle offset towards the target as well and would move the robot motors accordingly to match the angle.

The Arduino Uno also continuously kept track of temperature changes in the surrounding environment using the temperature sensor. If the temperature dropped below 15 degrees Celsius, it would activate the speaker and play an audio file that said "Brr, it's cold".

The LED matrix display is programmed with 8 bytes, each one corresponding to a row. By default, the Arduino Uno sets the matrix to display a smiley face. However, in situations, like when the robot tips over or when it is too cold outside, the matrix displays a sad face instead.

The Arduino Uno is connected to the Arduino Nano, through I2C, which is where all the speaker activation and audio files are processed.

5.5 Information Stored and Calculated on Nano

The Arduino Nano has an SD card module and speaker connected to it, along with the I2C connection to the Arduino Uno. The microcontroller receives a number from the UNO that corresponds to what file it should play. The files stored in the SD card are WAVE audio files. Upon receiving the number of the file that has to be played, the code in the Arduino Nano allows it to obtain this file from the SD card and send it to the speaker to be played.

6. Additional Improvements

Several improvements to the device could be made that would improve its functionality. Improvements in the object detection software could help avoid tracking drops when similar objects enter closer to the camera. Other improvements such as making the frame lighter would improve mobility, speed and increase the perception of the robot shivering when the haptic motor is active. More input from the user could also be taken to provide better control of the robot.

7. Evaluation of Performance

All the functionalities that were proposed, both required and functional, were achieved. Some issues such as the robot's excessive weight, made it difficult for the robot to move, and we were not able to implement a separate portable power source, although it was not a goal we had initially identified. Despite the challenges along the way, we were able to achieve what we set out to do.

Conclusion

This project, although ambitious at first, proved to be a great learning opportunity for the entire team. By combining our knowledge of microcontrollers, hardware peripheral components, machine learning and other tools, we were able to prototype a pet rock that could act as a children's toy. As mentioned above, there are several improvements that could be made to this robot. However, given the initial proposal, our understanding of the problem a few months ago, and our limited time constraint, we were able to meet the goals that had been set out for this project.