Sprint contribution Report

Sprint 1

To do

-Come up with a design problem relating to the “box” theme

-Choose a design idea

-Figure out components

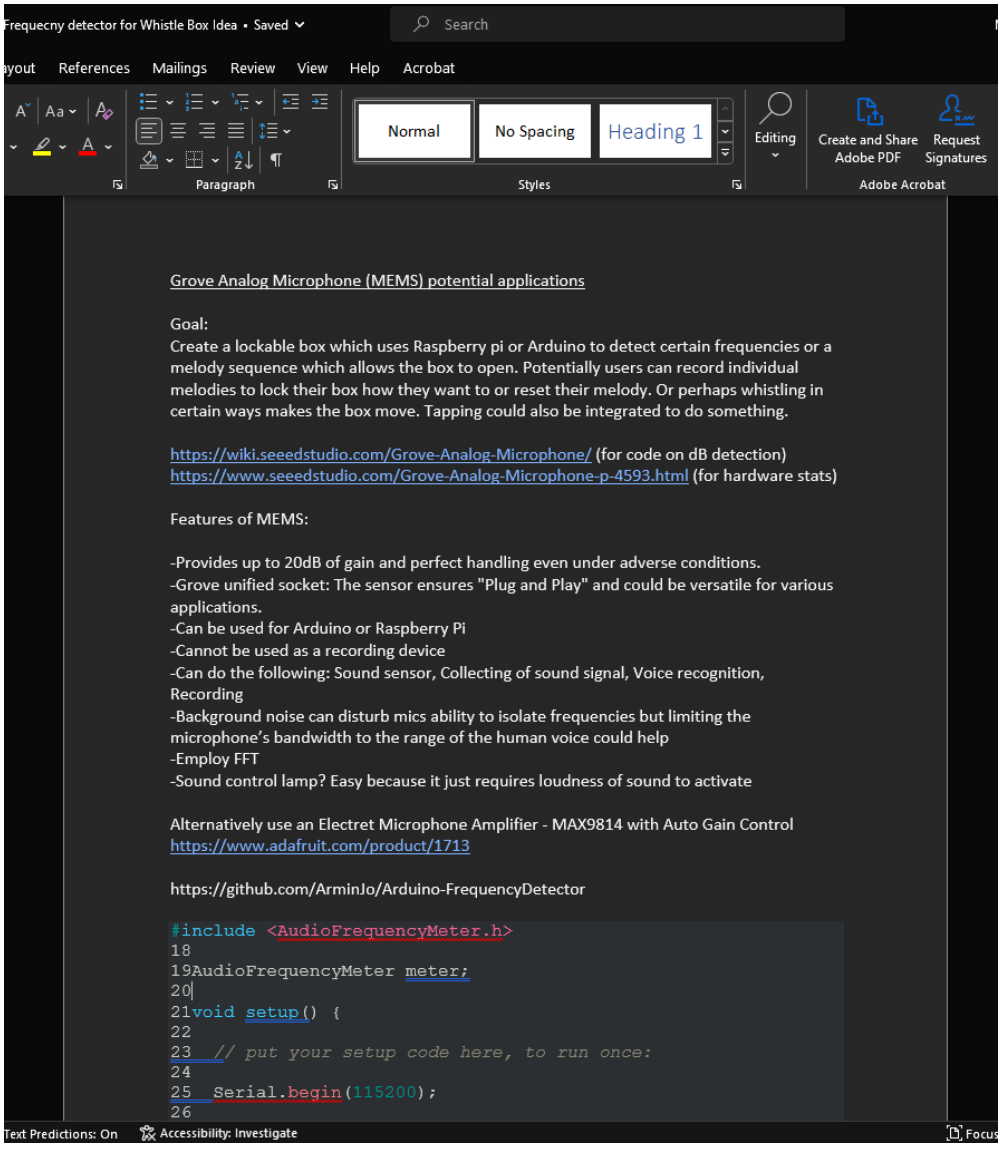
-Choose a target audience

My contribution

Came up with the whistle box idea in group ideation stage:

My idea was to create a novelty box that could be opened by whistling a certain frequency. I came up with a sensor to use, and found examples of whistling projects for Arduino to control lights such as the following: https://github.com/ArminJo/Arduino-FrequencyDetector

Word document with box idea info



Helped decide on a design idea during our designated group meeting:

During our meeting I realised the fealty of my idea, it was overly complex to code and beyond the capabilities of an Arduino and the sensors we had access to. There was also some tension when choosing between two of the other group members ideas, the fridge security detector, and the moving desktop pet. To consolidate this, I listened to both group members ideas and presented them with the limitations of their ideas and how they would overcome this. However, as a group we went with a moving car idea instead as it was more attractive to us and seemed more practical and achievable.

Retrospective 1 takeaways:

Group members were too attached to their individual ideas, should've listened to each other more and chose ideas that would work the best not personal preferences. In the group meeting this was improved as we considered the pros and cons of each idea and decided on one together. Assigning tasks did not go well and our target audience was difficult to specify.

Sprint 2

To do

-Figure out how the car will move forward

-Figure out how the wheels change direction

-Figure out if two motorised rear wheels have enough power to carry phone, wallet, water bottle

-Figure out what components need to be used

My contribution

Found and studied code for dexter GoPiGo3 motor driver:

We had been given a motor by the teacher that could be used for our car and studied a manual on how to code it, but our group ended up not going with that type of motor.

I helped ideate solution to user problem with our car idea:

Answered an in-class questionnaire to come up with basic functionality of our user following car idea "A moving robot that detects a person's heat signature and follows behind them, can carry a water bottle. The 'head' has sensors, and it turns to continually detect where the user is. The car will turn."

Also came up with component list by answering questionnaire and defined data types and ranges of sensors/components. As seen in showcase 2.

In-class questionnaire

Data Mapping

Q : What type of mapping approach will you use to transform the data to be output?

A: One possible mapping approach that could be used is a simple proportional control algorithm. In this approach, the distance between the owner and the baggage is measured using the ultrasonic and infrared sensors, and this distance is used to calculate a control signal that determines how fast and in what direction the baggage should move.

Q: What type of data does your output device require?

A: Heat and distance

Q: What is the range of the output data (min/max)?

A: No range, the device either moves or doesn't

Connections and Cables

Q:What needs to be connected to what?

Q:What type of connectors will you use? Wires,

Q:How big are the connectors and how far out do they extend normally?

Q:Will they need to be inserted and removed, and if so how much extra space will be needed to insert them?

Power

Q:What power does everything need?

Q: everything be run on 5V? (Which is what the Arduino pushes out?)

NO

Q:Will you need to have a battery on the inside or can it be placed on the outside? Outside How will you recharge it? Unknown

Answering the in-class questionnaire and helping ideating solutions to problems with idea

Device info

A moving robot that detects a person's heat signature and follows behind them, can carry a water bottle. The 'head' has sensors, and it turns to continually detect where the user is. The car will turn.

Components:

Ultrasonic Sensor

Range: 2cm - 3.5m distance converted to 0 – 1023

Infrared Sensor

https://www.auselectronicsdirect.com.au/infra-red-obstacle-sensor-module-for-arduino-proje?gclid=CjwKCAjw5pShBhB_EiwAvmnNVxbFcPd6Jd6xfBcPsj3LFF_5oc9baVe4AEy6kVM0PWlYTOSByjQ0MxoCKL8QAvD_BwEdetection

Range: Unknown heat value converted to 0 - 1023 with a 2- 40cm range Effective Angle: 35 degrees

OR

https://www.seeedstudio.com/Grove-Infrared-Emitter.html
https://wiki.seeedstudio.com/Grove-Infrared_Receiver/

Range: 10 meters, receiver just detects the emitter no extra data has to be collected

DC Motor and Arduino wheel x2

Servo motor turns the pole that holds the sensors Chassis to hold everything.

Q: What is their size and how can you position and fasten them? Small, with tape

Are there any limitations or complexities that need to be incorporated?

A: Need to make the infrared and ultrasonic only detect 1-2 meters so it doesn't pick up surrounding heat signals (unless were using the receiver type). Also placing it in a tube reduces the angle of detection so it focuses on one object. Is there any latency? Data Reduction Is data reduction necessary for your sensors? (ie. is your data too big)

Q: What summary data parameters can be generated from your sensor? Does any data reduction introduce latency? How much, and is it acceptable? Data Mapping What type of mapping approach will you use to transform the data?

A: Unknown

Q: Are you looking for triggers or for controls? Triggers Is your mapping linear or logarithmic/exponential?

A: Unknown

What is the range of input data (min/max)?

2cm - 3.5m distance converted to 0 - 1023 for ultrasonic and an unknow heat value for the infrared

Processing Needed

After you have data in the correct format, what processing will you need to do?

Are there any complex algorithms that need to be tested (eg face recognition etc.)

Need an algorithm that makes the device move in the direction of a heat source if it detects a heat signature that is equal to human heat temperature. Also need an algorithm that detects distances away from an object and stops if its too close. And finally one that limits the range of the ultrasonic and infrared

Retrospective 2 takeaways:

Better assigning of tasks and more communication would've saved me time trying to learn that manual.

Sprint 3

To do

- The basic frame structure of the cart. Battery, wheel, Arduino board, and l298N motor driver installed.

- Getting the automated box prototype working

- Code for the three components: Flame sensor, automated box, and obstacle avoidance example code

My contribution

Came up with the idea of using a grove flame sensor to potentially detect people via infrared light.

Found example code for the L298N motor driver.

Found example code and tested flame sensor and decided against it because it only works with objects emitting high levels of infrared light

I thoroughly researched into infrared sensors and presented one to the group to purchase. A lot of considerations went into it, including the range of the sensor.

Retrospective 3 takeaways:

Finally decided on the design idea and to go through with it despite previous apprehensions that it wouldn't work. Unfortunately, the car was not working with example code once assembled.

Sprint 4

To do

- Basic functionality in the car via code

- Build the main functions of the car

- Fix function which uses a switch to turn the car on and off

- Test obstacle avoiding function

- Decide on the sensor being mounted on a turning “head” platform turned by servo motor

My contribution

Fixed the car:

I took the car to a Jaycar store to diagnose the problem and got it working by replacing the existing 12v battery pack with the correct 9v one. I then implemented the correct wiring configuration for motor as it was wrong before. After testing the car with basic example code, the wheels were working.

Edited questionnaire:

After the research design I edited them to make them more detailed, relevant and to get more insight.

Created first instance of working code for our idea:

I tested the obstacle avoidance example that group members found online and then, after many iterations, adapted the code to pursue obstacles instead of avoiding obstacles and additionally, to stop in front of them. I then created a turning function to find objects and ended up with a working prototype using the ultrasonic sensor for detection. I then met with the group bringing the car to test it at university campus and showed it worked. Below are the code iterations from planning to the final working code.

Retrospective 4 takeaways:

Switch for on/off function was not created, more group collaboration needed.

First code iteration (ultrasonic only)

Obstaclepursuitcode.ino

```ino
#include "Ultrasonic.h"

#define ULTRASONIC_SENSOR_TRIG 9
#define ULTRASONIC_SENSOR_ECHO 8
#define MAX_REGULAR_MOTOR_SPEED 75
#define MAX_MOTOR_ADJUST_SPEED 150
#define DISTANCE_TO_CHECK 50
#define STOP_DISTANCE 15

//Right motor
int enableRightMotor=5;
int rightMotorPin1=8;
int rightMotorPin2=9;

//Left motor
int enableLeftMotor=6;
int leftMotorPin1=10;
int leftMotorPin2=11;

void rotateMotor(int rightMotorSpeed, int leftMotorSpeed);
Ultrasonic ultrasonic(7);

void setup()
{
  // put your setup code here, to run once:
  pinMode(enableRightMotor,OUTPUT);
  pinMode(rightMotorPin1,OUTPUT);
  pinMode(rightMotorPin2,OUTPUT);

  pinMode(enableLeftMotor,OUTPUT);
  pinMode(leftMotorPin1,OUTPUT);
  pinMode(leftMotorPin2,OUTPUT);

  rotateMotor(0,0);
}

void loop()
{


  int distance = ultrasonic.MeasureInCentimeters();

  if (distance > 0 && distance <= DISTANCE_TO_CHECK)
  {
    // Pursue object
    rotateMotor(MAX_REGULAR_MOTOR_SPEED, MAX_REGULAR_MOTOR_SPEED);
  }
    if (distance <= STOP_DISTANCE)
    {
      // Stop motors
      rotateMotor(0, 0);
      delay(500);
    }

    else if (distance > DISTANCE_TO_CHECK && distance <= 100){
        rotateMotor(-MAX_REGULAR_MOTOR_SPEED, MAX_REGULAR_MOTOR_SPEED);
        distance = ultrasonic.MeasureInCentimeters();
        if (distance > STOP_DISTANCE && distance <= DISTANCE_TO_CHECK) {
          return;
        }
      else {
      if (distance > DISTANCE_TO_CHECK && distance <= 100){
        rotateMotor(MAX_REGULAR_MOTOR_SPEED, -MAX_REGULAR_MOTOR_SPEED);
        distance = ultrasonic.MeasureInCentimeters();
        if (distance > STOP_DISTANCE && distance <= DISTANCE_TO_CHECK) {
          return;
        }
      }
    }
}}

```

```ino
void rotateMotor(int rightMotorSpeed, int leftMotorSpeed)
{
  if (rightMotorSpeed < 0)
  {
    digitalWrite(rightMotorPin1, LOW);
    digitalWrite(rightMotorPin2, HIGH);
  }
  else if (rightMotorSpeed >= 0)
  {
    digitalWrite(rightMotorPin1, HIGH);
    digitalWrite(rightMotorPin2, LOW);
  }

  if (leftMotorSpeed < 0)
  {
    digitalWrite(leftMotorPin1, LOW);
    digitalWrite(leftMotorPin2, HIGH);
  }
  else if (leftMotorSpeed >= 0)
  {
    digitalWrite(leftMotorPin1, HIGH);
    digitalWrite(leftMotorPin2, LOW);
  }

  analogWrite(enableRightMotor, abs(rightMotorSpeed));
  analogWrite(enableLeftMotor, abs(leftMotorSpeed));
}
```

Second code iteration

```
#include "Ultrasonic.h"

#define ULTRASONIC_SENSOR_TRIG 9
#define ULTRASONIC_SENSOR_ECHO 8
#define MAX_REGULAR_MOTOR_SPEED 75
#define MAX_MOTOR_ADJUST_SPEED 150
#define DISTANCE_TO_CHECK 15

//Right motor
int enableRightMotor=5;
int rightMotorPin1=8;
int rightMotorPin2=9;

//Left motor
int enableLeftMotor=6;
int leftMotorPin1=10;
int leftMotorPin2=11;

void rotateMotor(int rightMotorSpeed, int leftMotorSpeed);
Ultrasonic ultrasonic(7);

void setup()
{
  // put your setup code here, to run once:
  pinMode(enableRightMotor,OUTPUT);
  pinMode(rightMotorPin1,OUTPUT);
  pinMode(rightMotorPin2,OUTPUT);

  pinMode(enableLeftMotor,OUTPUT);
  pinMode(leftMotorPin1,OUTPUT);
  pinMode(leftMotorPin2,OUTPUT);

  rotateMotor(0,0);
}
```

```
void loop()
{

  int distance = ultrasonic.MeasureInCentimeters();

  //If distance is within 15 cm then adjust motor direction as below
  if (distance > 0 && distance < DISTANCE_TO_CHECK)
  {
    //Stop motors
    rotateMotor(0, 0);
    delay(500);

    //Reverse motors
    rotateMotor(-MAX_MOTOR_ADJUST_SPEED, -MAX_MOTOR_ADJUST_SPEED);
    delay(200);

    //Stop motors
    rotateMotor(0, 0);
    delay(500);

    //Rotate car to left
    rotateMotor(-MAX_REGULAR_MOTOR_SPEED, MAX_REGULAR_MOTOR_SPEED);
    delay(500);

    //Read left side distance using ultrasonic sensor
    int distanceLeft = ultrasonic.MeasureInCentimeters();

    if (distanceLeft == 50 )
    {
      rotateMotor(MAX_MOTOR_ADJUST_SPEED, MAX_MOTOR_ADJUST_SPEED);
      delay(200);
    }

    else {
    //Return to centre then rotate car to right
    rotateMotor(MAX_REGULAR_MOTOR_SPEED, -MAX_REGULAR_MOTOR_SPEED);
    rotateMotor(MAX_REGULAR_MOTOR_SPEED, -MAX_REGULAR_MOTOR_SPEED);
    delay(500);

    //Read right side distance using ultrasonic sensor
    int distanceRight = ultrasonic.MeasureInCentimeters();


    if (distanceRight == 50 )
    {
      rotateMotor(MAX_MOTOR_ADJUST_SPEED, MAX_MOTOR_ADJUST_SPEED);
      delay(200);
    }

    rotateMotor(0, 0);
    delay(200);
  }
  }
  else
  {
    rotateMotor(MAX_REGULAR_MOTOR_SPEED, MAX_REGULAR_MOTOR_SPEED);
  }
}
```

Third code iteration (First time implementing “void turnCar” function to add more parameters when car turns to achieve basic functionality)

```
#include <Adafruit_MLX90614.
#include "Ultrasonic.h"
#include <Wire.h>

#define ULTRASONIC_SENSOR_TRIG 9
#define ULTRASONIC_SENSOR_ECHO 8
#define MAX_REGULAR_MOTOR_SPEED 85
#define MAX_MOTOR_ADJUST_SPEED 130
#define DISTANCE_TO_CHECK 50
#define DISTANCE_TO_STOP 15
#define TURN_DURATION 500 // duration for turning in milliseconds

//Right motor
int enableRightMotor=5;
int rightMotorPin1=8;
int rightMotorPin2=9;

//Left motor
int enableLeftMotor=6;
int leftMotorPin1=10;
int leftMotorPin2=11;

void rotateMotor(int rightMotorSpeed, int leftMotorSpeed);
void turnCar();
Ultrasonic ultrasonic(7);
Adafruit_MLX90614 mlx = Adafruit_MLX90614();

void setup()
{
  Serial.begin(9600);
  // put your setup code here, to run once:
  pinMode(enableRightMotor,OUTPUT);
  pinMode(rightMotorPin1,OUTPUT);
  pinMode(rightMotorPin2,OUTPUT);

  pinMode(enableLeftMotor,OUTPUT);
  pinMode(leftMotorPin1,OUTPUT);
  pinMode(leftMotorPin2,OUTPUT);

  rotateMotor(0,0);
}
```

```
void setup()
{
  Serial.begin(9600);
  // put your setup code here, to run once:
  pinMode(enableRightMotor,OUTPUT);
  pinMode(rightMotorPin1,OUTPUT);
  pinMode(rightMotorPin2,OUTPUT);

  pinMode(enableLeftMotor,OUTPUT);
  pinMode(leftMotorPin1,OUTPUT);
  pinMode(leftMotorPin2,OUTPUT);

  rotateMotor(0,0);
}

void loop()
{
  int distance = ultrasonic.MeasureInCentimeters();


  if (distance > DISTANCE_TO_CHECK)
  {
    turnCar();
    }

  else if (distance > 0 && distance <= DISTANCE_TO_STOP)
  {
    rotateMotor(0, 0);
          delay(1000);
          return;
  }

  else if (distance > DISTANCE_TO_STOP && distance <= DISTANCE_TO_CHECK)
  {
    rotateMotor(MAX_REGULAR_MOTOR_SPEED, MAX_REGULAR_MOTOR_SPEED);
    while (true) {
      int distance = ultrasonic.MeasureInCentimeters();
      if (distance > 0 && distance <= DISTANCE_TO_STOP) {
        rotateMotor(0, 0);
         delay(1000);
         return;
      }
      else if (distance > DISTANCE_TO_CHECK)
        turnCar();
        break;
  }
}
}
```

```
void turnCar() {
//Rotate car to left
    rotateMotor(-MAX_MOTOR_ADJUST_SPEED, MAX_MOTOR_ADJUST_SPEED);
    delay(500);

    //Read left side distance using ultrasonic sensor
    int distance = ultrasonic.MeasureInCentimeters();

    if (distance > DISTANCE_TO_STOP && distance <= DISTANCE_TO_CHECK)
    {
      rotateMotor(MAX_REGULAR_MOTOR_SPEED, MAX_REGULAR_MOTOR_SPEED);
      while (true) {
      int distanceLeft = ultrasonic.MeasureInCentimeters();
      if (distanceLeft > 0 && distanceLeft <= DISTANCE_TO_STOP) {
          rotateMotor(0, 0);
           delay(1000);
           return;
      }
    }
    }
      else {
    //Rotate car to right
    rotateMotor(MAX_MOTOR_ADJUST_SPEED, -MAX_MOTOR_ADJUST_SPEED);
    delay(500);

    //Read right side distance using ultrasonic sensor
    int distance = ultrasonic.MeasureInCentimeters();

    if (distance > DISTANCE_TO_STOP && distance <= DISTANCE_TO_CHECK)
    {
      rotateMotor(MAX_REGULAR_MOTOR_SPEED, MAX_REGULAR_MOTOR_SPEED);
      while (true) {
      int distanceLeft = ultrasonic.MeasureInCentimeters();
      if (distanceLeft > 0 && distanceLeft <= DISTANCE_TO_STOP) {
          rotateMotor(0, 0);
           delay(1000);
           return;
      }
```

Also added a while statement whilst going forward to ensure target was still in front of the car (otherwise car stops and returns to loop)

Code testing controlling movement with Infrared

MLXMOVEMENTTESTV3.ino

```ino
#include <Adafruit_MLX90614.h>
#include <Wire.h>

#define MAX_REGULAR_MOTOR_SPEED 130
#define MAX_MOTOR_ADJUST_SPEED 130

// Right motor
int enableRightMotor = 5;
int rightMotorPin1 = 8;
int rightMotorPin2 = 9;

// Left motor
int enableLeftMotor = 6;
int leftMotorPin1 = 10;
int leftMotorPin2 = 11;

Adafruit_MLX90614 mlx = Adafruit_MLX90614();

void setup() {
  Serial.begin(9600);
  while (!Serial);
  Serial.println("Adafruit MLX90614 test");

  if (!mlx.begin()) {
    Serial.println("Error connecting to MLX sensor. Check wiring.");
    while (1);
  }

  pinMode(enableRightMotor, OUTPUT);
  pinMode(rightMotorPin1, OUTPUT);
  pinMode(rightMotorPin2, OUTPUT);

  pinMode(enableLeftMotor, OUTPUT);
  pinMode(leftMotorPin1, OUTPUT);
  pinMode(leftMotorPin2, OUTPUT);

  rotateMotor(0, 0);
}
void loop() {
  int objectTemp = mlx.readObjectTempC();

  Serial.print("Object Temp: ");
  Serial.print(objectTemp);
  Serial.println(" *C");

  if (objectTemp >= 24) {
    // Perform action when temperature is above 24
    // For example, rotate the motor
    rotateMotor(MAX_REGULAR_MOTOR_SPEED, MAX_REGULAR_MOTOR_SPEED);
  } else {
    // Temperature is below 24, stop the motor
    rotateMotor(0, 0);
  }

  delay(1000);
}

void rotateMotor(int rightMotorSpeed, int leftMotorSpeed) {
  if (rightMotorSpeed < 0) {
    digitalWrite(rightMotorPin1, LOW);
    digitalWrite(rightMotorPin2, HIGH);
  } else if (rightMotorSpeed >= 0) {
    digitalWrite(rightMotorPin1, HIGH);
    digitalWrite(rightMotorPin2, LOW);
  }

  if (leftMotorSpeed < 0) {
    digitalWrite(leftMotorPin1, LOW);
    digitalWrite(leftMotorPin2, HIGH);
  } else if (leftMotorSpeed >= 0) {
    digitalWrite(leftMotorPin1, HIGH);
    digitalWrite(leftMotorPin2, LOW);
  }

  analogWrite(enableRightMotor, abs(rightMotorSpeed));
  analogWrite(enableLeftMotor, abs(leftMotorSpeed));
}
```

Final iteration with both infrared and ultrasonic

URSUECODE.ino

```ino
#include <Adafruit_MLX90614.h>
#include "Ultrasonic.h"
#include <Wire.h>

#define ULTRASONIC_SENSOR_TRIG 9
#define ULTRASONIC_SENSOR_ECHO 8
#define MAX_REGULAR_MOTOR_SPEED 160
#define MAX_MOTOR_ADJUST_SPEED 165
#define DISTANCE_TO_CHECK 50
#define DISTANCE_TO_STOP 15
#define TURN_DURATION 500

// Right motor
int enableRightMotor = 5;
int rightMotorPin1 = 8;
int rightMotorPin2 = 9;

// Left motor
int enableLeftMotor = 6;
int leftMotorPin1 = 10;
int leftMotorPin2 = 11;

void rotateMotor(int rightMotorSpeed, int leftMotorSpeed);
void turnCar();
Ultrasonic ultrasonic(7);
Adafruit_MLX90614 mlx = Adafruit_MLX90614();

void setup() {
  Serial.begin(9600);
  while (!Serial);

  mlx.writeEmissivity(1.0);

  Serial.println("Adafruit MLX90614 test");

  if (!mlx.begin()) {
    Serial.println("Error connecting to MLX sensor. Check w
    while (1);
  }

  pinMode(enableRightMotor, OUTPUT);
  pinMode(rightMotorPin1, OUTPUT);
  pinMode(rightMotorPin2, OUTPUT);

  pinMode(enableLeftMotor, OUTPUT);
  pinMode(leftMotorPin1, OUTPUT);
  pinMode(leftMotorPin2, OUTPUT);

  rotateMotor(0, 0);
}
```

```ino
void turnCar() {
//Rotate car to left
    rotateMotor(-MAX_MOTOR_ADJUST_SPEED, MAX_MOTOR_ADJUST_SPEED);
    delay(930);

    //Read left side distance using ultrasonic sensor
  int objectTemp = mlx.readObjectTempC();
  int distance = ultrasonic.MeasureInCentimeters();

    if (objectTemp >= 24 && distance > 0 && distance <= DISTANCE_TO_CHECK)
    {
      rotateMotor(MAX_REGULAR_MOTOR_SPEED, MAX_REGULAR_MOTOR_SPEED);
      while (true) {
      int objectTemp = mlx.readObjectTempC();
      int distance = ultrasonic.MeasureInCentimeters();
      if (objectTemp < 24 || distance > 0 && distance <= DISTANCE_TO_STOP || distance > DISTANCE_TO_CHECK) {
          rotateMotor(0, 0);
          delay(1000);
          return;
      }
    }
    }
      else {
    //Rotate car to right
    rotateMotor(MAX_MOTOR_ADJUST_SPEED, -MAX_MOTOR_ADJUST_SPEED);
    delay(930);

    //Read right side distance using ultrasonic sensor
      int turningrightobjectTemp = mlx.readObjectTempC();
      int distanceRight = ultrasonic.MeasureInCentimeters();

    if (objectTemp >= 24 && distance > 0 && distance <= DISTANCE_TO_CHECK)
    {
      rotateMotor(MAX_REGULAR_MOTOR_SPEED, MAX_REGULAR_MOTOR_SPEED);
      while (true) {
      int objectTemp = mlx.readObjectTempC();
      int distance = ultrasonic.MeasureInCentimeters();
      if (objectTemp < 24 || distance > 0 && distance <= DISTANCE_TO_STOP || distance > DISTANCE_TO_CHECK) {
          rotateMotor(0, 0);
          delay(1000);
          return;
      }
    }
      }
}
}
```

In this code I changed conditionals from && to || in the while statements to make sure all cases were covered in terms of the car stopping pursuing and returning to the start of the loop

```ino
void loop() {
  int objectTemp = mlx.readObjectTempC();
  int distance = ultrasonic.MeasureInCentimeters();

  Serial.print("Object Temp: ");
  Serial.print(objectTemp);
  Serial.println(" *C");

  if (objectTemp < 24 || distance > DISTANCE_TO_CHECK)
  {
    turnCar();
    }

  else if (objectTemp >= 24 && distance > DISTANCE_TO_STOP && distance <= DISTANCE_TO_CHECK)
  {
    rotateMotor(MAX_REGULAR_MOTOR_SPEED, MAX_REGULAR_MOTOR_SPEED);
    while (true) {
      int objectTemp = mlx.readObjectTempC();
      int distance = ultrasonic.MeasureInCentimeters();
      if (objectTemp < 24 || distance > 0 && distance <= DISTANCE_TO_STOP || distance > DISTANCE_TO_CHECK) {
          rotateMotor(0, 0);
          delay(1000);
          return;
      }
    //  else if (distance > DISTANCE_TO_CHECK && objectTemp < 24)
      //  turnCar();
      // break;
  }
}

}
```

Initial written "code logic plan" that led to the third iteration of code

1. Measures dis
2. If >50 & <100 turn left/Right.
else If something within 15cm STOP
else if >0 & <50cm
3B.
GO
Measure distance

3A. Measure left/Right
if Something in front GO
if nothing in front STOP delay
and Return to beginning of loop

4A. Whilst GO, measure
if within 15cm Stop
if within 50 GO
Measure
if within 15 cm stop
if within 50 GO

1. Measures distance ✓
2. [IF distance > Distance to check & < 100 (turn left/Right) Go to step 3.]
[else if distance >0 & < Distance to stop (Stop) (delay 1000) (return)]
[else if distance > Distance to stop & < Distance to check (GO);
while (true): Measure distance and if distance is >0 & < Distance to stop
(Stop) (delay 1000) (return/break)

3. Void turn left/Right [rotate car to the left, measure
if distance > Distance to stop & < Distance to check [(GO);
while (true); Measure and if distance >0 & < Distance to stop
(stop) (delay 1000) (return/break)]
else
[Return to centre then go right, measure
if distance > Distance to stop & < Distance to check [(GO);
while (true): Measure and if distance >0 & < Distance to
stop (stop) (delay 1000) (return/break)]
]
"else (return)" ???

Turn function

Turns left
- If its greater than 15cm and less than 50cm Go until 15cm or less then stops
- Otherwise (else if less than 15cm or greater than 50cm) it turns Right then if its greater than 15cm and less than 50 it goes until 15cm or less
- else if (within 0-15cm) Stop, delay and return
else (return to beggining loop)

Sprint 5

To do:

-Improve speed variables and stop functions in the code

-3D print box

-User testing

-Designing questionnaire

My contribution:

Added infrared component: After receiving the component via post I edited the code to get the infrared working and it was successful in working in tandem with the ultrasonic.

Performed user testing on User 2: At university I performed testing on user 2 and recorded the data.

Edited code based on user testing: Based on user testing I attempted to edit the code further to improve the prototype but was unsuccessful.

Retrospective 5 takeaways:

Coding went well, succeeded in creating a complete and working prototype according to our plan.