**Team 6 Report: Richard Wan, Kevin Zheng, Sam Park, Sartaj Chowdhury**

**Introduction:**

This project was motivated by the problem of rescuing victims of disaster zones that are too dangerous for human rescue teams to traverse. To this end we designed a robot that can navigate unknown environments and locate potential survivors in need of assistance. More specifically we implemented a wall following algorithm to navigate the unknown environment. For this task we created several subsystems to divide the original large goal into more manageable components. These consisted generally of locomotion, navigation and target detection. These systems wanted to navigate the environment with reasonable speed and efficiency and be able to detect a hot object that would be an analog for a human survivor, and perform an action after finding a survivor. The solution had to be inexpensive to produce and operate. For these goals we set general expectations that the solution would complete a general course in less than 5 minutes and detect human body temperature of 98.6 degrees Fahrenheit.

**Design Process:**

Using the provided components made dividing the subsystems fairly straightforward which we delineated into the following subsystems.

Heat Detection - Thermopile, Thermopile circuit, buzzer

* This subsystem will detect a hot bottle of water, sending a signal to the logic subsystem and then sounding a buzzer

Locomotion - motors, wheels, motor driver

* This subsystem will allow for the robot to move around based on input from the wall detection from the logic subsystem

Wall Detection - Ultrasonic sensors, Ultrasonic sensors brackets

* This subsystem will use sensors to measure the distance from the side and front of the robot to their respective walls and send input to the logic subsystem for decision making.

Logic - multiplexer, Arduino, Arduino code

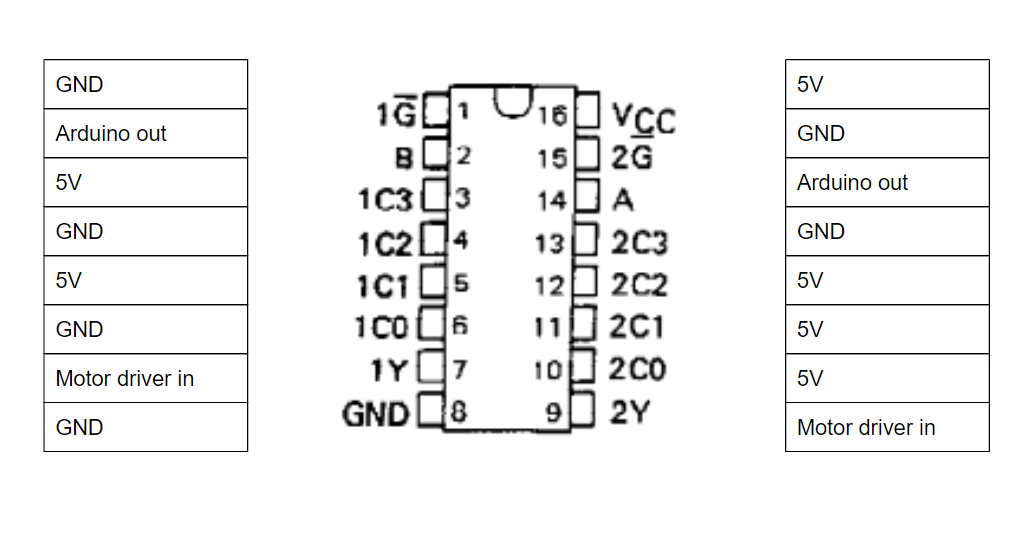
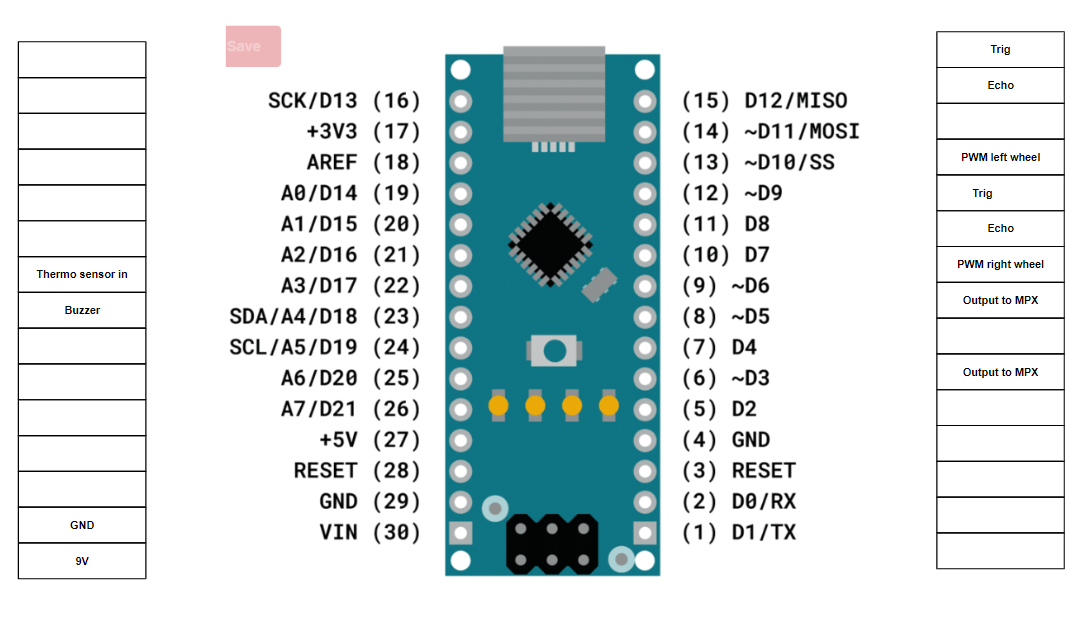
* This subsystem will be in charge of interpreting the information from the wall detection and heat detection system and send decisions via voltage signals to the locomotion subsystem.

Structure - Chassis, tape

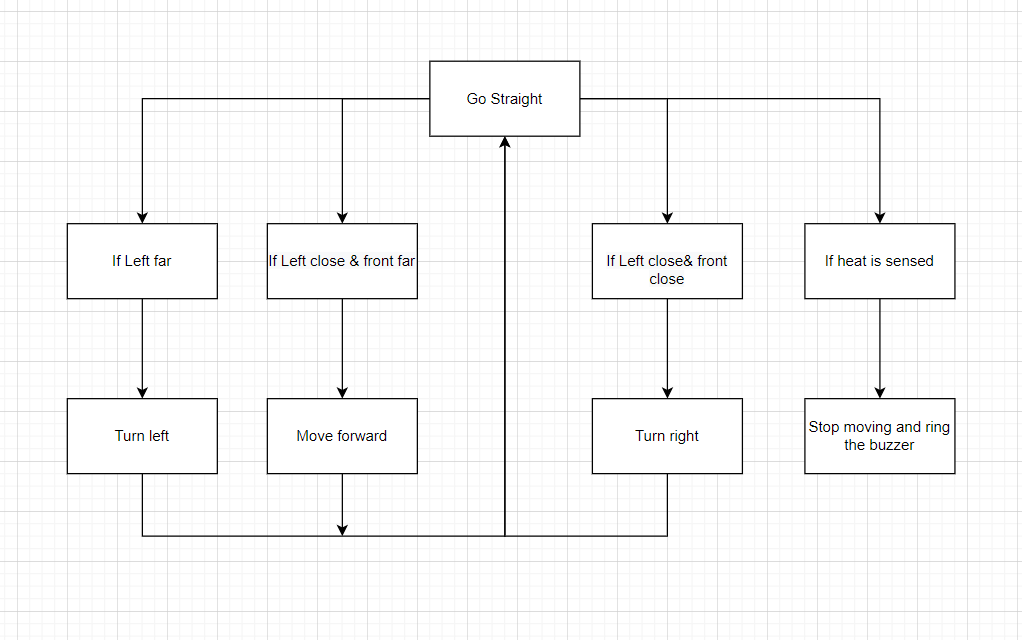
* This subsystem will be the main body of the robot and will carry all the different components from the subsystems.

Power - wires, Batteries

* This subsystem provides the necessary battery power to the electrical components of the subsystems via wires.

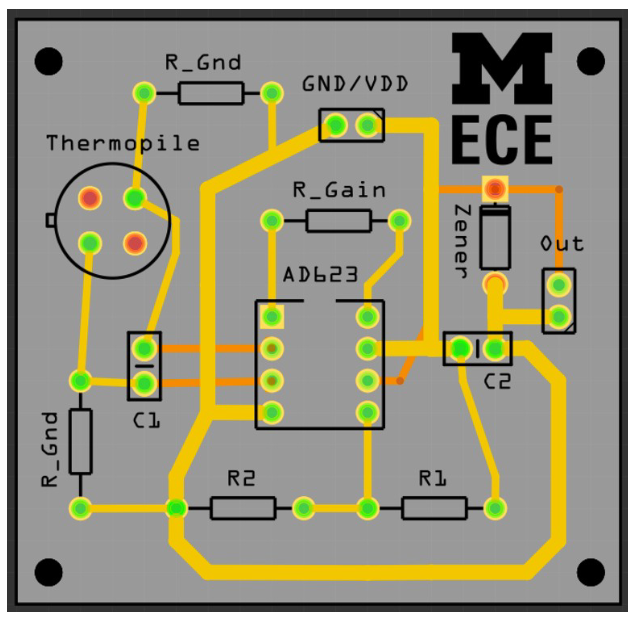
**Multiplexer wiring:Arduino wiring diagram:**

**Robot and code flow chart:**



**Heat Detection Subsystem:**

**Thermopile Circuit diagram**

****

This amplifier circuit is powered by our 5 volt rail on the breadboard and amplifies the signal from our thermopile sensor and outputs the amplified voltage into the arduino analogread(). This allows us to set a threshold voltage to detect ranges of temperature and the value we chose to detect human body temperature was 270 on the analogread(). Once a value higher than 270 is detected by the pin, the PWM pins to the motor driver output 0 to stop the motors and the buzzer pin sends 5V to the buzzer to ring it. This proved very flexible and was able to be tuned more sensitively depending on the heat of the target. This was made extra sensitive for the final test where it successfully detected the bottle. The heat detection system was completed using a circuit from the appropriate lab and carried over into the adapted arduino iteration used on the robot. Testing was done using the DMM and detecting high voltage output individually from the circuit output. The next verification step was allowing the robot to use the circuit on its own to detect hot objects. And was attached using tape to the front of the robot and wired into the 5V rail of our breadboard for powering the opamp as well as direct output into the arduino to transmit its information.

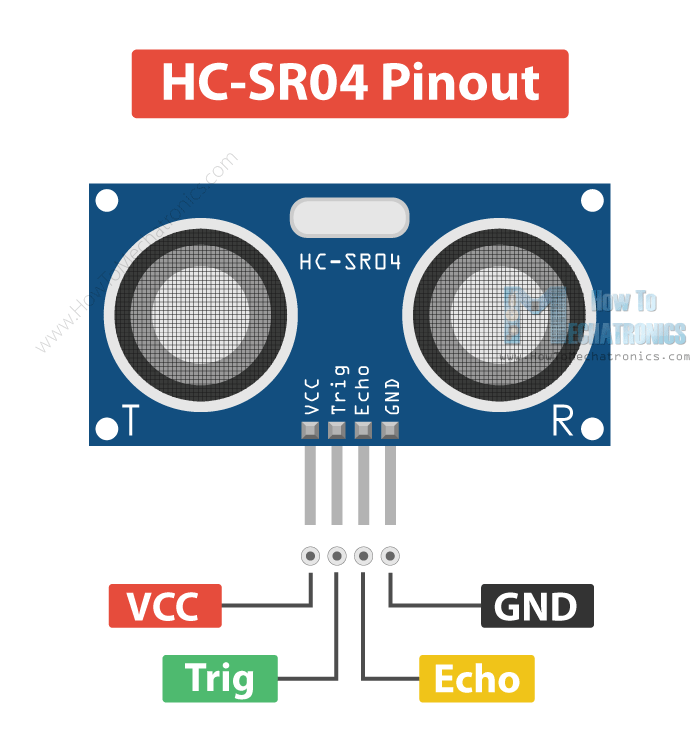
**Locomotion**

This subsystem allows for the robot to move around based on input from the wall detection from the logic subsystem. The motor driver receives a signal from the multiplexer output that is determined by the logic set by us in the truth table above. Wire connections were soldered onto the leads of the motors and connected into the motor driver output terminals for both wheels. The DC power supply was used to ensure that the wheels would operate in the proper direction. Our general goal was to have the solution complete the general course in less than 5 minutes which was achieved.

**Structure Subsystem**

The general structure of the robot is largely determined by the chassis as it holds all the components of the different subsystems together and allows for the robot to retain its structural integrity. Most of the components are attached to the chassis via electrical tape so as to not interfere with the electrical components of the subsystems. The general weight of the robot was distributed in such a way that the heaviest components would be closer to the center of the chassis or at least near the center wheel to not have the robot lose its balance by tipping too far on the front or back sides as that would cause the robot to not move as intended.

**Wall Detection Subsystem**



The wall detection subsystem makes use of the ultrasonic sensors that are on the front and left side of the robot in order to provide detection of the respective walls. Each of the sensors are affixed to a bracket that is attached directly to the robot and wired to the circuit such that the VCC, Trig, and the Echo pins on the sensor are connected to the arduino. The sensor requires a 5V power supply (anything past 5V would cause the sensor to cease functioning), so its VCC pin was connected to the arduino’s 5V output pin rather than the 9V rail on the circuit board. The Trig and Echo pins are the outputs of the sensors and they were connected to the arduino’s digital pins so that it could read in those values to later calculate distance values in the logic subsystem. The sensor accuracy was tested via arduino code through output on the terminal window and we found that the sensors were off by at most 3cm which we determined as sufficient enough for our robot.

**Logic**

1. The front and left sensor send data to the arduino
2. Depending on the sensor distance, determine if the two sensors are close or far
3. If the left sensor is far, turn left and move forward a little
4. Left sensor is close and
   1. If front sensor is close, turn right
   2. If front sensor is far, go straight with a bit of right turn biase
5. If the thermophile detects a hot object, set the pwm to 0 for both wheels, send 5V to the buzzer, and delay indefinitely.

**Logic Debugging**

Some corrections we made with each update were:

1. Slowed down the robot to allow sensors to detect and react to the walls.
2. Added a right bias to insure the robot will not collide with a straight wall.
3. Adjusted the proportions for the left turn which allows the robot to better follow the wall during left turns.
4. Increase the left wheel’s pwm during right turns so that the robot does not get caught by the corner in the situation where it does not turn enough to catch the next wall.

**Power**

This subsystem provides the necessary battery power to the electrical components of the subsystems via wires. The power from the battery, 9 volts, powers the motor driver and the arduino nano in our circuit. The individual components are powered over 5 volts from the arduino nano into the multiplexer selection and output, sonar sensors, and buzzer. Finally, power from the motor driver powers the both left and right motors. These connections were individually tested using the DMM to ensure proper power delivery and no connections were shorted.

**Components**

Heat Detection - Thermopile, Thermopile circuit, buzzer

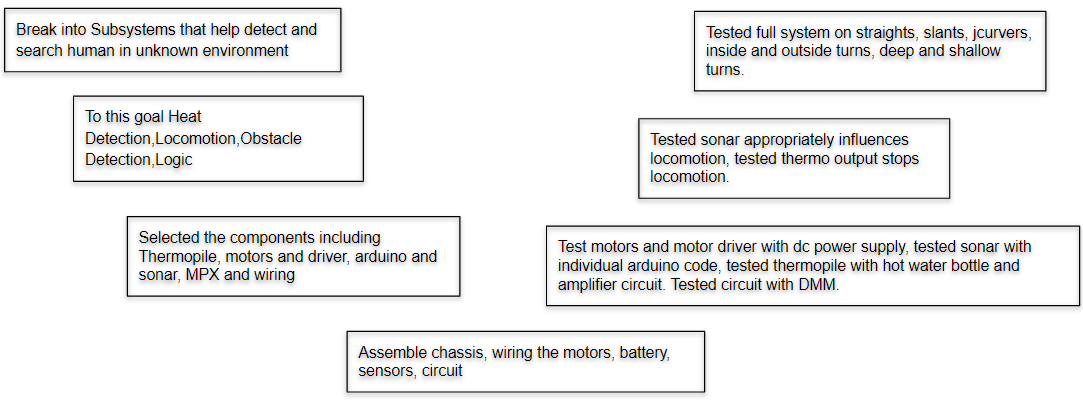
Locomotion - motors, wheels, motor driver

Wall Detection - Ultrasonic sensors, Ultrasonic sensors brackets

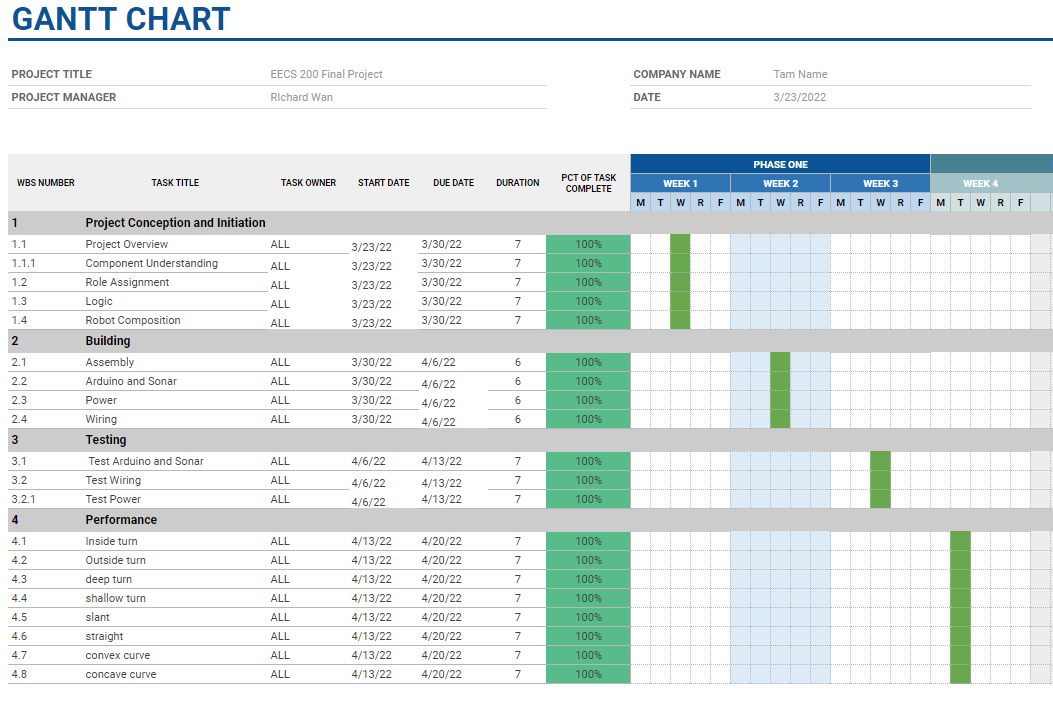
Logic - multiplexer, Arduino, Arduino code

Structure - Chassis, tape

Power - wires, Batteries

**V diagramTruth table:**

| SF | SL | I1L | I2L | I1R | I2R |
| --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 |

****

**What trade-offs were made?**

Some trade-offs that were made include slowing down the robot to allow the sensors enough time to detect and react to obstacles. Another trade-off is we need a right bias when going straight so that the robot does not collide with straight or slightly slanted walls making the robot susceptible to collision for shallow turns. Finally our solution can only handle generic right turns (eg. 90 degrees), which does not account for extremely sharp corners (eg. 180 degrees) .

**Results: Provide an overview of the competition-day performance. Did your robot perform as you expected it to? What went well? What did not go well? Ideas on how to improve the performance.**

On competition day, our robot performed not exactly the way we expected it to. During our test runs on the maze before competition, the robot was able to navigate through the entire maze without any mistakes and successfully sensed the hot water bottle on the first try but it did not have consistent results for later tests. The detection systems had a couple of close calls where they narrowly actuated sharp left and right turns to avoid obstacle collisions. What went well was that it had the ability to do all the tasks correctly and could stop and sound a buzzer after detecting the hot bottle of water which gave us time to tweak the code to improve consistency. However, what did not go well was that it still had many inconsistencies and required certain start positions to be able to navigate the maze without crashing. Some improvements that could have been made would be to reduce the robots speed so that the sensors could catch up, change the delay values in the code and maybe find alternate methods for delay as it fixed problems but also may have caused a lot more as well due to sensors being ignored during delays.

**References**

Arduino nano spec sheet - <https://www.makerguides.com/arduino-nano/>

Multiplexer datasheet - <https://www.unicornelectronics.com/ftp/Data%20Sheets/74153.pdf>

Motor Driver spec sheet - <https://www.instructables.com/Tutorial-for-MD-L298-Motor-Driver-Module/>

**Appendix: include your code**

// ---------------------------------------------------------------------------

// Example NewPing library sketch that pings 3 sensors 20 times a second.

// ---------------------------------------------------------------------------

#include <NewPing.h>

#define SONAR\_NUM 2 // Number of sensors.

#define MAX\_DISTANCE 200 // Maximum distance (in cm) to ping.

#define front 3

#define left 5

#define pwmleftwheel 6

#define pwmrightwheel 9

#define buzzer 4

NewPing sonar[SONAR\_NUM] = { // Sensor object array.

NewPing(12, 11, MAX\_DISTANCE), // Each sensor's trigger pin, echo pin, and max distance to ping.

NewPing(8, 7, MAX\_DISTANCE)

};

void setup() {

Serial.begin(115200); // Open serial monitor at 115200 baud to see ping results.

pinMode(front, OUTPUT);

digitalWrite(front, LOW);

pinMode(left, OUTPUT);

digitalWrite(left, LOW);

analogWrite(pwmleftwheel, 40);

analogWrite(pwmrightwheel, 30);

}

void loop() {

delay(75);

for (uint8\_t i = 0; i < SONAR\_NUM; i++) { // Loop through each sensor and display results.

delay(50); // Wait 50ms between pings (about 20 pings/sec). 29ms should be the shortest delay between pings.

Serial.print(i);

Serial.print("=");

Serial.print(sonar[i].ping\_cm());

Serial.print("cm ");

}

int front\_dist = sonar[0].ping\_cm();

int left\_dist = sonar[1].ping\_cm();

int left\_thresh = 21;

int front\_thresh = 21;

bool front\_close = (front\_dist <= front\_thresh) && (front\_dist > 1);

bool left\_close = (left\_dist <= left\_thresh) && (left\_dist > 1 );

if (front\_close){

digitalWrite(front,HIGH);

Serial.print("CLOSE111 ");

}

if (!front\_close){

digitalWrite(front,LOW);

Serial.print("FAR111 ");

}

if (!left\_close){

digitalWrite(left,LOW);

Serial.print("FAR");

}

if (left\_close){

digitalWrite(left,HIGH);

Serial.print("CLOSE");

}

if(left\_close && front\_close){

analogWrite(pwmleftwheel, 85);

}

if(left\_close && !front\_close){

analogWrite(pwmleftwheel,38);

analogWrite(pwmrightwheel, 31);

}

if(!left\_close){

digitalWrite(front,LOW);

analogWrite(pwmrightwheel, 46);

digitalWrite(left,LOW);

delay(75);

analogWrite(pwmleftwheel, 38);

analogWrite(pwmrightwheel, 27);

//go straight

digitalWrite(left,HIGH);

delay(75);

}

int thermoVal = analogRead(3);

if(thermoVal > 270){

analogWrite(pwmrightwheel, 0);

analogWrite(pwmleftwheel, 0);

analogWrite(buzzer, 255);

delay(100000000);

}

Serial.println();

}