Yufan Liu
Serena Wee
ME102b

The Buggy: Mini self navigation cart

1. **Opportunity**

The opportunity for the self-navigating cart is to introduce young children to basic robotics and automation principles. It serves as a practical tool for understanding spatial awareness and navigation algorithms, while also offering a hands-on experience in simple programming and mechanical design, thus fostering early technical skills development children-friendly. Our target audience is children and teenagers from age 6 to age 15.

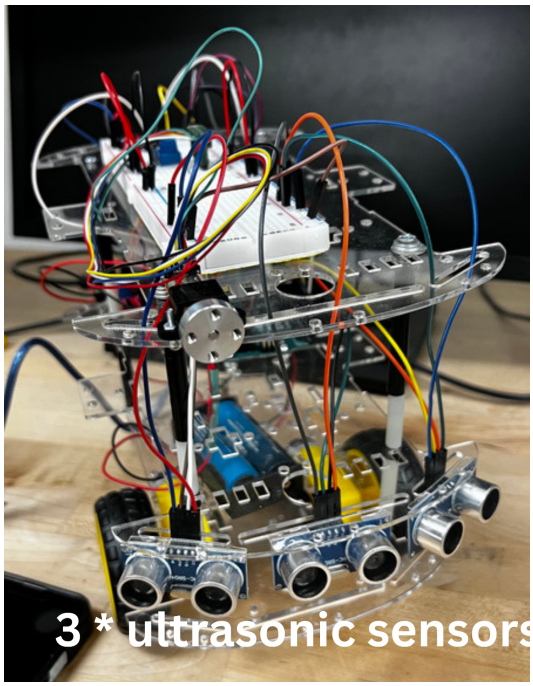
2. **The high-level strategy:** for the self-navigating cart is to develop a multifunctional educational tool that engages young children in STEM learning, with several initial desired functionalities.

- basic self-navigation with obstacle avoidance
- Simple and direct control using buttons and potentiometer for instant response
- We added features such as button control to begin movement and 3 sensors instead of only having one sensor

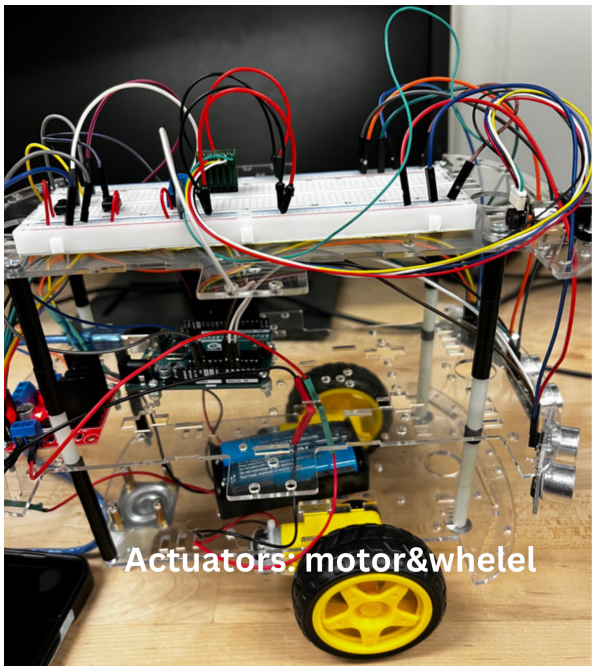
Achieved:

- The cart now includes more sophisticated sensor algorithms for obstacle detection to deal with spiky sensor noise (an outlier reading every 10 to 15 seconds depending on setting)
- Immediate stop when it is about to hit obstacles in front of the cart

3. Photos



Yufan Liu
Serena Wee
ME102b



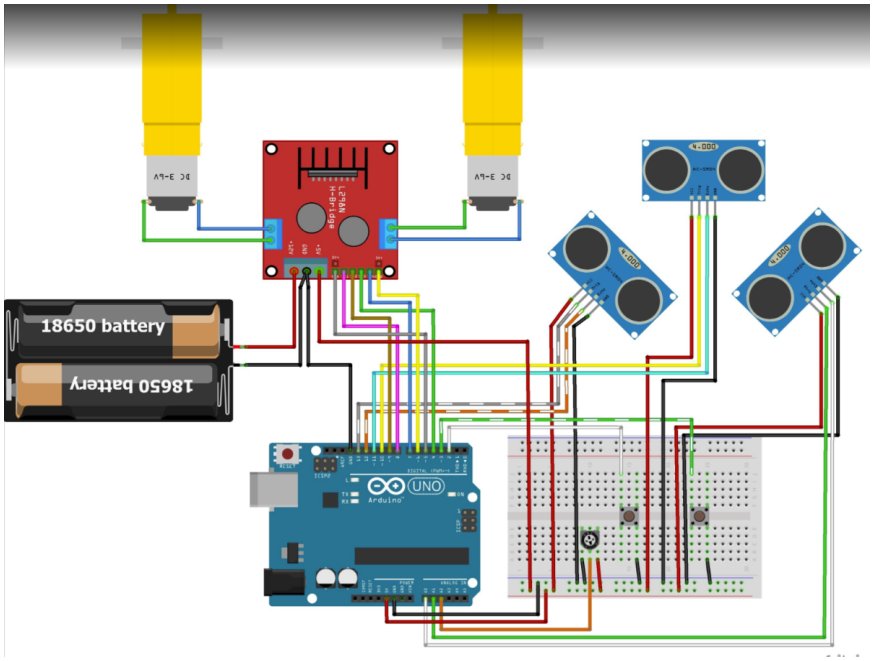
3. Torque and Load Calculations

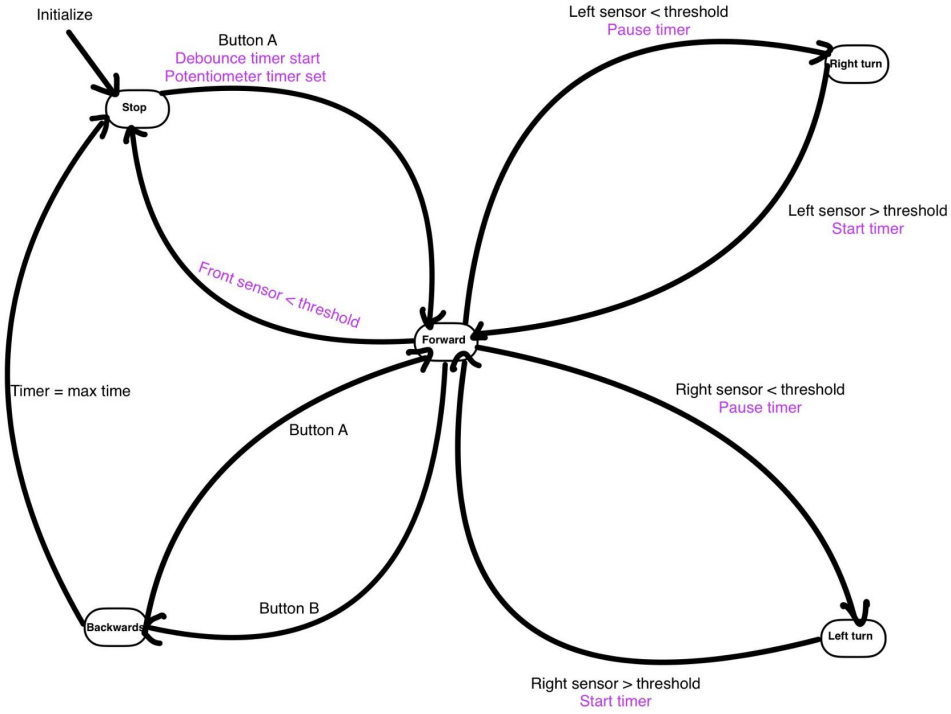
- Stall Torque (6V): 0.8kg.cm = 0.078 Nm
- Gear Ratio (GR): 1:48
- Wheel radius (r) = 0.031m
- Calculated with assumed efficiencies
    - Motor efficiency = 70%
    - Gear train efficiency = 70%
    - Wheel-to-ground friction efficiency = 70%

$$
\begin{aligned}
T\_output &= T\_stall * GR \\
&= 0.078 * 48\,Nm \\
&= 3.74\,Nm \\
TotalForce &= 2 * T / r \\
&= 2 * 3.74\,Nm / 0.031m \\
&= 250\,N \\
Cart\ Load &= TotalForce / g \\
&= 25kg \\
\eta &= \eta motor * \eta gear * \eta wheel \\
&= 0.35 \\
Adjusted\ Load &= CartLoad * \eta \\
&= 8.75\ \text{kg}
\end{aligned}
$$

Yufan Liu
Serena Wee
ME102b

4. Circuit Diagram and State Machine Diagram





5. **Reflection**

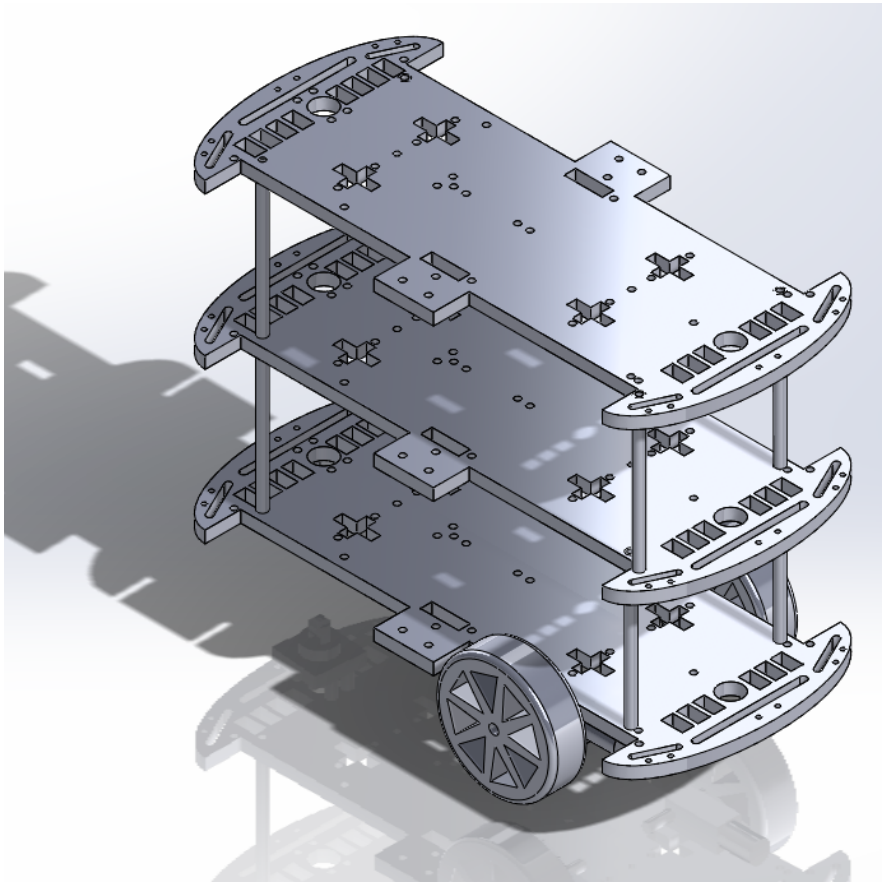
Our group found that frequent testing of the behavior of our self-navigating cart was crucial. In iterative testing, we uncovered unexpected behaviors and undesirable state transitions that were not apparent in our original state machine design. This at the end helped us design and achieve a robust state machine.

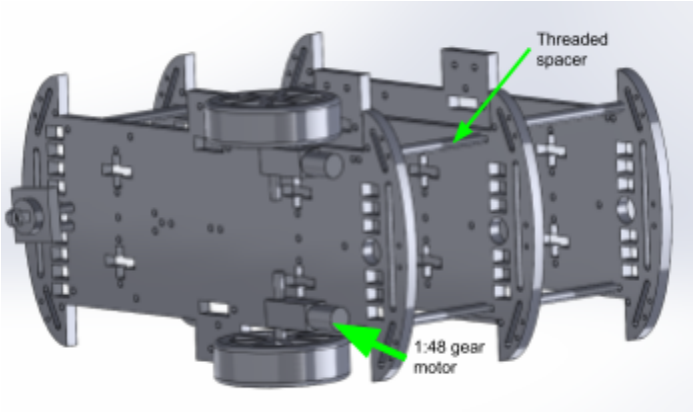
Yufan Liu
Serena Wee
ME102b

Strategies that worked well for our group is constant communication and being up front about issues you faced

Appendix

Yufan Liu
Serena Wee
ME102b





Yufan Liu
Serena Wee
ME102b

A	B	C	D	E	F
part number	Item name	Quantity	unit price	Estimated cost	Link to website
1	YIKESHU 4WD 2 Layer Smart Robot Car Chassis Kit with Speed Encoder Battery Box for Kids Teens DIY	1	$21.49	$21.49	https://www.amazon.com/YIKESHU-Smart-Chassis-E
2	20 Pcs 6mm 2 Pin Momentary Tactile Tact Push Button Switch Through Hole Breadboard Friendly for Panel PCB	1	$5.99	$5.99	https://www.amazon.com/MakerSpot-Momentary-Tact
3	ELEGOO 5PCS HC-SR04 Ultrasonic Module Distance Sensor Compatible with Arduino UNO MEGA Nano Robot XBee ZigBee	1	$8.99	$8.99	https://www.amazon.com/ELEGOO-HC-SR04-Ultraso
4	Qunqi 2Packs L298N Motor Drive Controller Board Module Dual H Bridge DC Stepper For Arduino	1	$8.99	$8.99	https://www.amazon.com/Qunqi-2Packs-Controller-St
5	PNGKNYOCN 18650 Battery Clip, 2 Slots 3.7V DIY Battery Storage Box in Series Plastic Batteries Case YOUCHENG for 18650 Battery with Connect Lead	1	$8.99	$8.99	Amazon.com: PNGKNYOCN 18650 Battery Clip, 2 Sl
6	Arduino Uno REV3 [A000066]	1	$27.60	$27.60	https://a.co/d/fTjofDW
7	ELEGOO 120pcs Multicolored Dupont Wire 40pin Male to Female, 40pin Male to Male, 40pin Female to Female Breadboard Jumper Ribbon Cables Kit Compatible with Arduino Projects	1	$6.98	$6.98	https://a.co/d/4NeF2To
8	Electronics-Salon Black Nylon Round Spacer Assortment Kit, Not Threaded for M4 Screws, Plastic. OD 7mm, ID 4.1mm, L 2mm 4mm 5mm 6mm 8mm 10mm 12mm 16mm 18mm 21mm	1	$9.29	$9.29	https://a.co/d/ith2GrP
9	uxcell M4x20mm Flat Head Machine Screws Inner Hex Screw 304 Stainless Steel Fasteners Bolts 30Pcs	1	$6.99	$6.99	https://www.amazon.com/gp/aw/d/B0BNJMTQVX/?_e
10	6/32 threaded spacer 1 inch length	4	$ 1.19	$4.76	only in ACE store
11	6/32 spacer 1inch length	4	$0.45	$1.80	only in ACE store
12	6/32 philip head screw 3inch length	8	$0.35	$2.80	Hillman No. 6-32 X 3 in. L Combination Round Head Z
13	Pololu Micro Metal Gearmotor Bracket Pair – Black	1	$2.95	$2.95	https://www.pololu.com/product/989
14	Hex (Allen) Wrench 0.05 #: 1064	1	$0.75	$0.75	Pololu - Hex (Allen) Wrench 0.05"
15	Adafruit Pocket Screwdriver – Black	1	$1.50	$1.50	Adafruit Pocket Screwdriver - Black : ID 3284 : $1.50 :
16	Breadboard trim potentiometer	2	$1.25	$2.50	Breadboard trim potentiometer [10K] : ID 356 : $1.25 :
17	75:1 Micro Metal Gearmotor HP 6V with Extended Motor Shaft	2	$20.95	$41.90	https://www.pololu.com/product/2215

Yufan Liu
Serena Wee
ME102b

car_final_testingV2.ino

```
/****************** Ultrasonic module *********************/
#define AVOIDANCE_DISTANCE 25       // obstacle avoidance distance, when the ultrasonic s
#define AVOIDANCE_TURNING_TIME 200  // obstacle avoidance steering time [in milliseconds]

#define RUN_INTERVAL_TIME 200        // Run acquisition interval [in milliseconds]
#define COLLECTION_INTERVAL_COUNT 5  // interval difference
#define DISTANCE_INTERVAL_VALUE 10   // difference [in cm]

int TrigPin[] = { 13, 10, A1 };  // Ultrasonic Trig pin
int EchoPin[] = { 12, 11, A0 };  // Ultrasonic Echo pin

int SR04Num = sizeof(TrigPin) / sizeof(TrigPin[0]);  // Number of ultrasonic waves
struct SR04 {
  float newDistance = 200.0;  // New data, with default value
  float oldDistance = 200.0;  // old data, default value
  float Value = 200.0;        //
  int collectionCount = 0;    // Number of distance intervals
  unsigned long runTime = 0;  // Record the phase running time
};
SR04 sr04[sizeof(TrigPin) / sizeof(TrigPin[0])];
/*****************************************************************/
/************************* button ***********************/
#define buttonPin1 2  // Button 1 is connected to digital pin 2
#define buttonPin2 3  // Button 2 is connected to digital pin 3
volatile int buttonIsPressed = false;
/*****************************************************************/

/************************* potentiometer ***********************/
#define POT_PIN A2              // potentiometer pin
#define ADJUST_MAXIMUM_TIME 20  //   Adjust maximum time[in second]
#define ADJUST_MINIMUM_TIME 0   //   Adjust maximum time[in second]
unsigned long elapsedTime = 0;  //
/*****************************************************************/
/**************** L298N controls the Pin ***********************/
#define MOTOR_IN1 8              // left motor control pin 1
#define MOTOR_IN2 9              // left motor control pin 2
#define MOTOR_IN3 4              // Right motor control pin 1
#define MOTOR_IN4 7              // right motor control pin 2
#define enable1 5                // left motor speed control
#define enable2 6                // Right motor speed control
#define SPEED_STEP 5             // amount of steps for each adjusted speed
```

Yufan Liu
Serena Wee
ME102b

```
#define MOTORINIT_SPEED 500        // initial trolley speed
#define LEFT_SPEED_COMPENSATE 0   // Left motor adds compensation
#define RIGHT_SPEED_COMPENSATE 0  // right motor adds compensation

int motorSpeed = MOTORINIT_SPEED;  // Current speed of the storage trolley
/*************************************************/
/*********************** Serial Port Printing ************************************/
#define pintTime 1000          // Interval time for getting gyroscope data
bool statusResetFlag = false;  // State machine reset flag
bool runFlag = false;
bool directionFlag = false;
int allowRunningTime = 0;  // Allowed running time
int state = 0x00;         //initial state
int runTime = 0;                // Time already run [Unit: seconds]
unsigned long runlastTime = 0;  //
unsigned long adjustTime = 0;   // Record time information for adjusting state data

/*****************************************************************/
void setup() {
  delay(1000);
  // Start the serial port communication
  Serial.begin(9600);

  // Configure the button pins as input with the internal pull-up resistor enabled
  pinMode(buttonPin1, INPUT_PULLUP);
  pinMode(buttonPin2, INPUT_PULLUP);
  // Attach an interrupt to the button pins, the interrupt service routines (buttonPre
  attachInterrupt(digitalPinToInterrupt(buttonPin1), buttonPressed1, LOW);
  attachInterrupt(digitalPinToInterrupt(buttonPin2), buttonPressed2, LOW);
  // Set the motor control pin as the output
  pinMode(MOTOR_IN1, OUTPUT);
  pinMode(MOTOR_IN2, OUTPUT);
  pinMode(MOTOR_IN3, OUTPUT);
  pinMode(MOTOR_IN4, OUTPUT);

  // Set the speed control pin as the output
  pinMode(enable1, OUTPUT);
  pinMode(enable2, OUTPUT);
```

Yufan Liu
Serena Wee
ME102b

```
  // Ultrasonic initialization
  for (int i = 0; i < SR04Num; i++) {
    pinMode(TrigPin[i], OUTPUT);  // Set the pin output
    pinMode(EchoPin[i], INPUT);   // Set the pin input
  }
}

void loop() {

  switch (state) {
    case 0x00:  // initial state, Detect which button (front/back) is pressed
      {
        if (CheckForButtonPress() == 1)  // Forward
        {
          directionFlag = true;
          Serial.print("set forward time: ");
          Serial.print(allowRunningTime = map(1023 - analogRead(POT_PIN), 0, 1023, ADJU
          Serial.println("s");
          runlastTime = millis();
          runTime=0;
          state++;
        }
        if (CheckForButtonPress() == 2)  // Backward
        {
          directionFlag = false;
          Serial.print("set Back time: ");
          Serial.print(allowRunningTime = map(1023 - analogRead(POT_PIN), 0, 1023, ADJU
          Serial.println("s");
          adjustTime = millis();
          state++;
        }
      }
      break;
    case 0x01:  // Determine the direction of the vehicle
      {
        if (directionFlag == true) {
          moveForward();  // Move forward
          state++;
        } else {
          moveBackward();  // Move backward
```

Yufan Liu
Serena Wee
ME102b

```
        state = 0x07;
      }
    }
    break;
  case 0x02:  // Detect distance
    {
      for (int i = 0; i < SR04Num; i++) {
        srFilteredData(i);
      }
      state++;
    }
    break;
  case 0x03:  // Determine if the running time has been reached
    {
      state++;
    }
    break;
  case 0x04:  // Judge distance, whether to turn, if no need to turn, record running time
    {
      int minState = 0;
      for (byte i = 0; i < SR04Num; i++) {
        if (sr04[i].Value < AVOIDANCE_DISTANCE) {
          minState = i + 1;
          break;
        }
      }
      switch (minState) {
        case 1:  // Detected left side less than threshold, execute right turn
          {
            Serial.println(F("Detected left side less than threshold, executing right turn"));
            turnRight();
            adjustTime = millis();
            state++;
          }
          break;
        case 2:  // Detected backward less than threshold, execute stopMoving, enter init mode
          {
```

Yufan Liu
Serena Wee
ME102b

```
            Serial.println(F("Detected forward less than threshold, stopMoving"));
            stopMoving();
           // adjustTime = millis();
            state = 0x00;
            buttonIsPressed=0;
          }
          break;
        case 3:  // Detected right side less than threshold, execute left turn
          {
            Serial.println(F("Detected right side less than threshold, executing left turn"));
            turnLeft();
            adjustTime = millis();
            state++;
          }
          break;
        default:
          {
            state = 0x02;
            if (millis() - runlastTime > 1000) {
              runlastTime = millis();
              runTime++;
              if (runTime >= allowRunningTime) {
                state = 0x00;
                buttonIsPressed = 0;
                stopMoving();
                Serial.println(F("Running time reached, vehicle stops"));
              }
            }
          }
          break;
      }
    }
    break;
  case 0x05:  // Judge if the turning time has been reached
    {
      if (millis() - adjustTime > AVOIDANCE_TURNING_TIME)  // Turning time reached
      {
```

Yufan Liu
Serena Wee
ME102b

```
            Serial.println(F("Turning end, rejudge distance"));
            stopMoving();
            adjustTime = millis();  // Update waiting time
            state++;
          }
        }
        break;
      case 0x06:  // Re-acquire distance
        {
          for (int i = 0; i < SR04Num; i++) {
            srFilteredData(i);
          }
          if (millis() - adjustTime > 2000)  // Turning time reached
          {
            state = 0x01;
            Serial.println(F("Turning end, execute forward movement"));
          }
        }
        break;
      case 0x07:  // Judge if the backward time has been reached
        {
          if (millis() - adjustTime > allowRunningTime*1000UL)  // Backward time reache
          {
            printf(adjustTime);
            printf(allowRunningTime*1000UL);
            state = 0x00;
            stopMoving();
            Serial.println(F("Backward time reached, stop running"));
          }
        }
        break;
      default:
        {
        }
        break;
    }

    SerialPrint(pintTime);
}
```

Yufan Liu
Serena Wee
ME102b

```
/*Function action : moving forward
 *Function name : moveForward
 *Function entry parameter : None
 *Return value : No
 */
void moveForward() {
  digitalWrite(MOTOR_IN1, HIGH);
  digitalWrite(MOTOR_IN2, LOW);
  digitalWrite(MOTOR_IN3, HIGH);
  digitalWrite(MOTOR_IN4, LOW);
  analogWrite(enable1, constrain((motorSpeed + LEFT_SPEED_COMPENSATE), 0, 255));
  analogWrite(enable2, constrain((motorSpeed + RIGHT_SPEED_COMPENSATE), 0, 255));
}
/*Function action : backward
 *Function name : moveBackward
 *Function entry parameter : None
 *Return value : No
 */
void moveBackward() {
  digitalWrite(MOTOR_IN1, LOW);
  digitalWrite(MOTOR_IN2, HIGH);
  digitalWrite(MOTOR_IN3, LOW);
  digitalWrite(MOTOR_IN4, HIGH);
  // analogWrite(enable1, motorSpeed);
  // analogWrite(enable2, motorSpeed);
  analogWrite(enable1, constrain((motorSpeed + LEFT_SPEED_COMPENSATE), 0, 255));
  analogWrite(enable2, constrain((motorSpeed + RIGHT_SPEED_COMPENSATE), 0, 255));
}
/*Function action : turn left
 *Function name : turnLeft
 *Function entry parameter : None
 *Return value : No
 */
void turnLeft() {
  digitalWrite(MOTOR_IN1, HIGH);
  digitalWrite(MOTOR_IN2, LOW);
  digitalWrite(MOTOR_IN3, LOW);
  digitalWrite(MOTOR_IN4, HIGH);
  analogWrite(enable1, motorSpeed);
  analogWrite(enable2, motorSpeed);
}
```

Yufan Liu
Serena Wee
ME102b

```
void turnRight() {
  digitalWrite(MOTOR_IN1, LOW);
  digitalWrite(MOTOR_IN2, HIGH);
  digitalWrite(MOTOR_IN3, HIGH);
  digitalWrite(MOTOR_IN4, LOW);
  analogWrite(enable1, motorSpeed);
  analogWrite(enable2, motorSpeed);
}
/*Function action : stop
 *Function name : stopMoving
 *Function entry parameter : None
 *Return value : No
 */
void stopMoving() {
  digitalWrite(MOTOR_IN1, LOW);
  digitalWrite(MOTOR_IN2, LOW);
  digitalWrite(MOTOR_IN3, LOW);
  digitalWrite(MOTOR_IN4, LOW);
  analogWrite(enable1, 0);
  analogWrite(enable2, 0);
}
/*Function : Sfilter the ultrasonic distance
 *Function name : srFilteredData
 *Function entry parameters :
 -- -- -- -- -- -- -_ Index : the subscript of the ultrasound wave in the array
  *Return value : float
  */
void srFilteredData(int _index) {
  if (millis() - sr04[_index].runTime >= RUN_INTERVAL_TIME) {  // Judge the time interval
    // After processing, reset the recording phase time
    sr04[_index].runTime = millis();                                                    // at the
    sr04[_index].newDistance = getDistanceData(TrigPin[_index], EchoPin[_index]);       // Store t
    if (abs(sr04[_index].newDistance - sr04[_index].oldDistance) >= DISTANCE_INTERVAL_VALUE) {  // if the
      if (++sr04[_index].collectionCount >= COLLECTION_INTERVAL_COUNT) {                // More th
        sr04[_index].oldDistance = sr04[_index].newDistance;                            // Update
        sr04[_index].collectionCount = 0;                                               // Run res
      }
    } else {                                              // if the data changes within a fixed range, t
      sr04[_index].oldDistance = sr04[_index].newDistance;  // Update the data values
    }
    sr04[_index].Value = sr04[_index].oldDistance;  // Update the data
  }
```

Yufan Liu
Serena Wee
ME102b

```
/*Function function : Obtain the ultrasonic distance
  *Function name : getDistanceData
  *Function entry parameters :
  -- -- -- -- -- -- -_trigPin : Ultrasonic trig pin
  -- -- -- -- -- -- -_echoPin : Ultrasonic echo pin
 *Return value : float
*/
float getDistanceData(int _trigPin, int _echoPin) {
  float tempDistance;                                  // Temporary storage variable
  digitalWrite(_trigPin, LOW);                         // Set the low level of the pin
  delayMicroseconds(2);                                // delay 2 is subtle
  digitalWrite(_trigPin, HIGH);                        // Set the pin high level
  delayMicroseconds(10);                               // delay wait 10 subtle
  digitalWrite(_trigPin, LOW);                         // Set the low level of the pin
  tempDistance = pulseIn(_echoPin, HIGH) / 58.0;       // calculate the distance by the formula
  tempDistance = (int(tempDistance * 100.0)) / 100.0;  // converted to cm
  if (tempDistance < 0) {                              // Judge that the detection distance data
    tempDistance = 400;                                // equal to 400
  } else if (tempDistance > 400) {                     // if the detection distance data is grea
    tempDistance = 400;                                // equal to 400
  }
  return tempDistance;  // output feedback distance
}

/* Function: Set the interval for serial printing
 * Input parameters: SerialPrint
 *           _time: [Unit: milliseconds]
 * Return value: None
 */
void SerialPrint(unsigned long _time) {  // Function to obtain analog values
  static unsigned long lastTime = 0;     // Establish a static local variable to store the last
  if (millis() - lastTime >= _time) {    // Check if current time - last time >= sampling time
    lastTime = millis();                 // Update time
    Serial.print("left:");
    Serial.print(sr04[0].Value);
    Serial.print("\tmiddle:");
    Serial.print(sr04[1].Value);
    Serial.print("\tright:");
    Serial.println(sr04[2].Value);
  }
}
```

Yufan Liu
Serena Wee
ME102b

```
//Event Checker
int CheckForButtonPress() {
  if (buttonIsPressed == 1) {
    buttonIsPressed = 0;
    return 1;
  } else if (buttonIsPressed == 2) {
    buttonIsPressed = 0;
    return 2;
  } else {
    return false;
  }
}

// Interrupt service routines
void buttonPressed1() {
  // This function is called when button 1 is presse
  buttonIsPressed = 1;
  state=0;
}

void buttonPressed2() {
  buttonIsPressed = 2;
  state=0;
}
```