MARS ROVER

Project by: Nodari Francesco, Tommaso Agosti, Davide Wu and Alessio Wu

System basic workflow

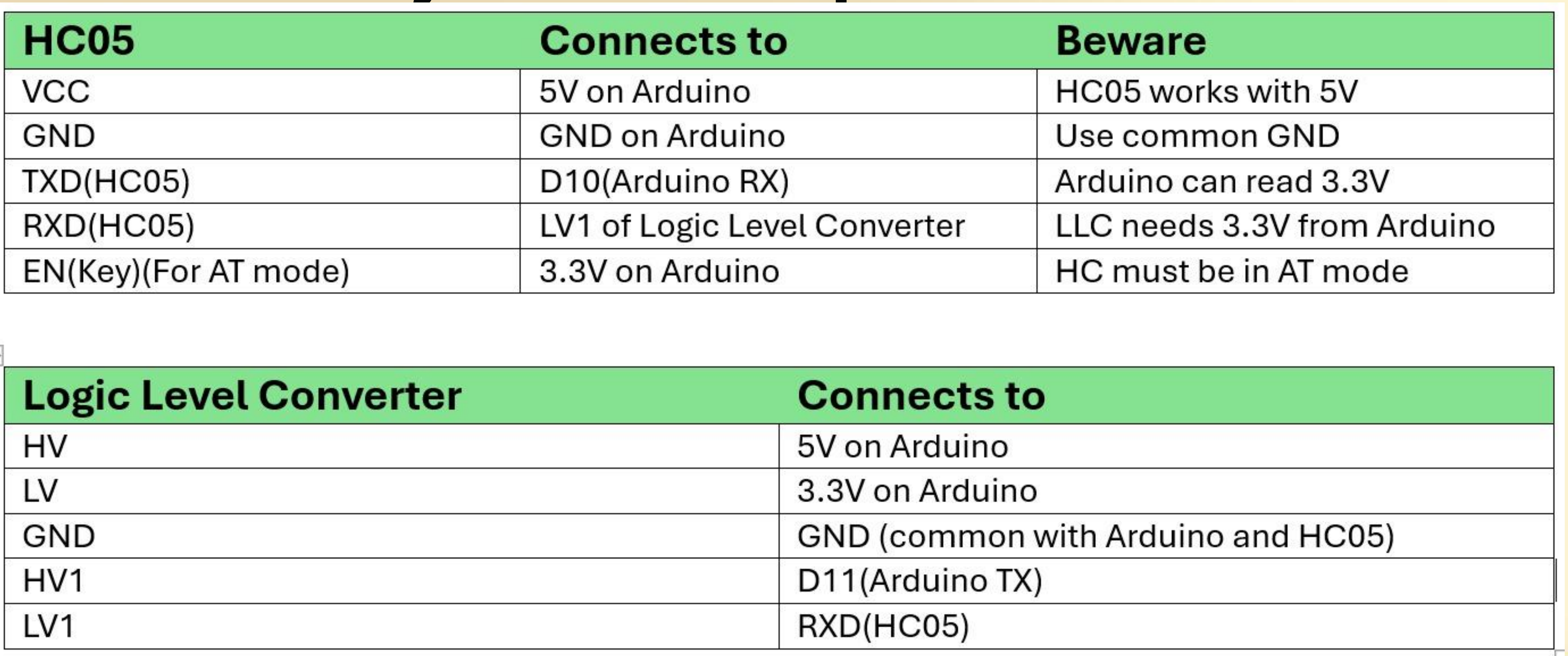
The hole robot was produced using a 3D printer.

Using the MSP432 boosterpack potentiometer we are able to control the robot movement, as a master; the informations are passed to the computer, via cable, and then to the hc05 via bluetooth.

The Arduino Uno, the slave, which will move accordingly to the commands.

An ultrasonic sensor, is mounted to detect the object near the rover and sends an interrupt if it is too close.

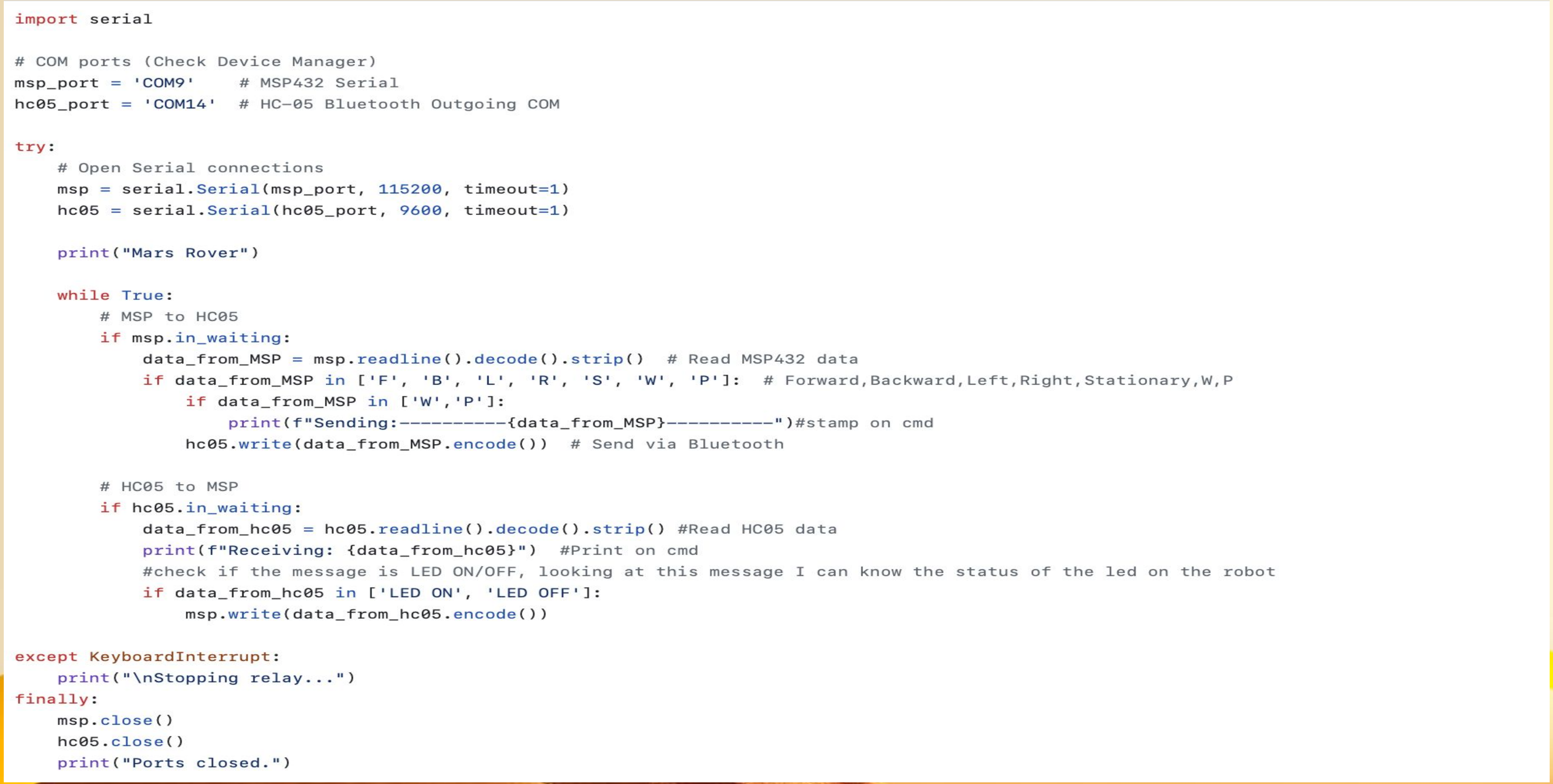
Project circuit pins scheme



HC05	Connects to	Beware
VCC	5V on Arduino	HC05 works with 5V
GND	GND on Arduino	Use common GND
TXD(HC05)	D10(Arduino RX)	Arduino can read 3.3V
RXD(HC05)	LV1 of Logic Level Converter	LLC needs 3.3V from Arduino
EN(Key)(For AT mode)	3.3V on Arduino	HC must be in AT mode

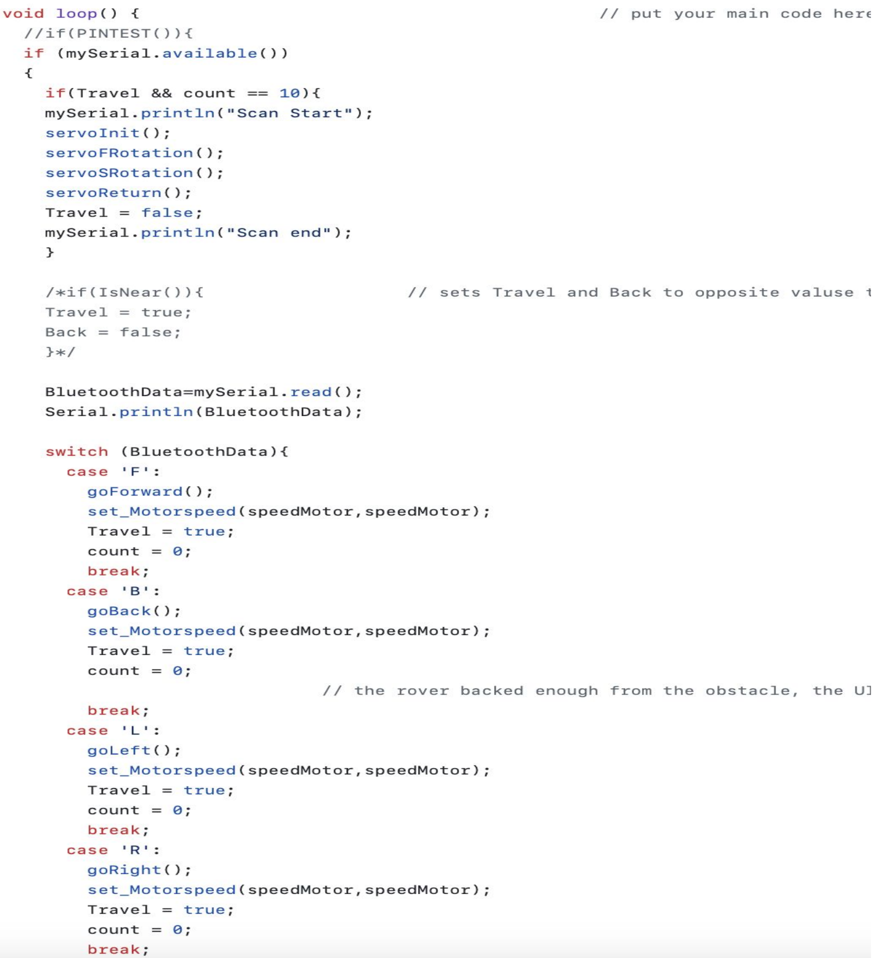
Logic Level Converter	Connects to
HV	5V on Arduino
LV	3.3V on Arduino
GND	GND (common with Arduino and HC05)
HV1	D11(Arduino TX)
LV1	RXD(HC05)

Bluetooth connection summary

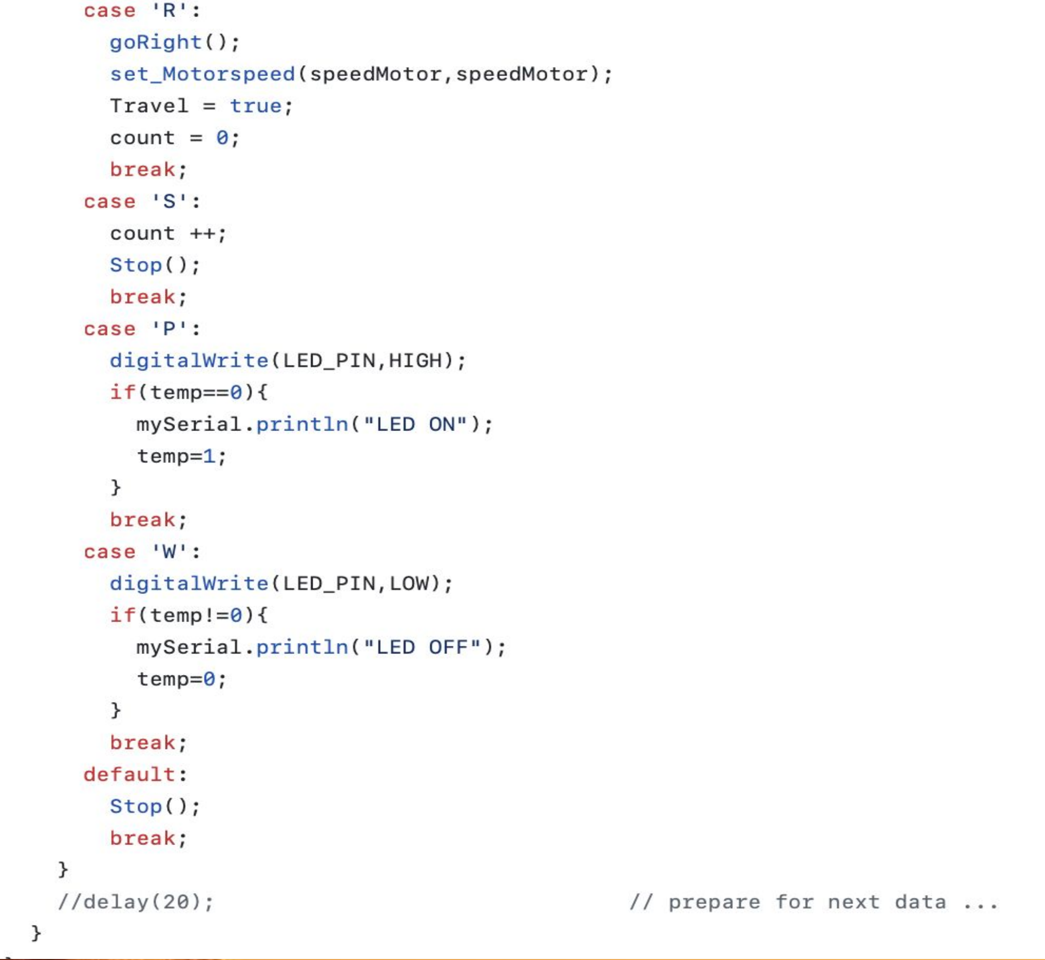


```python
import serial

# COM ports (Check Device Manager)
msp_port = 'COM9'     # MSP432 Serial
hc05_port = 'COM14'  # HC-05 Bluetooth Outgoing COM

try:
    # Open Serial connections
    msp = serial.Serial(msp_port, 115200, timeout=1)
    hc05 = serial.Serial(hc05_port, 9600, timeout=1)

    print("Mars Rover")

    while True:
        # MSP to HC05
        if msp.in_waiting:
            data_from_MSP = msp.readline().decode().strip()  # Read MSP432 data
            if data_from_MSP in ['F', 'B', 'L', 'R', 'S', 'W', 'P']:  # Forward,Backward,Left,Right,Stationary,W,P
                if data_from_MSP in ['W','P']:
                    print(f"Sending:----------{data_from_MSP}----------")#stamp on cmd
                hc05.write(data_from_MSP.encode())  # Send via Bluetooth

        # HC05 to MSP
        if hc05.in_waiting:
            data_from_hc05 = hc05.readline().decode().strip() #Read HC05 data
            print(f"Receiving: {data_from_hc05}")  #Print on cmd
            #check if the message is LED ON/OFF, looking at this message I can know the status of the led on the robot
            if data_from_hc05 in ['LED ON', 'LED OFF']:
                msp.write(data_from_hc05.encode())

except KeyboardInterrupt:
    print("\nStopping relay...")
finally:
    msp.close()
    hc05.close()
    print("Ports closed.")
```

Part of Rover’s movement

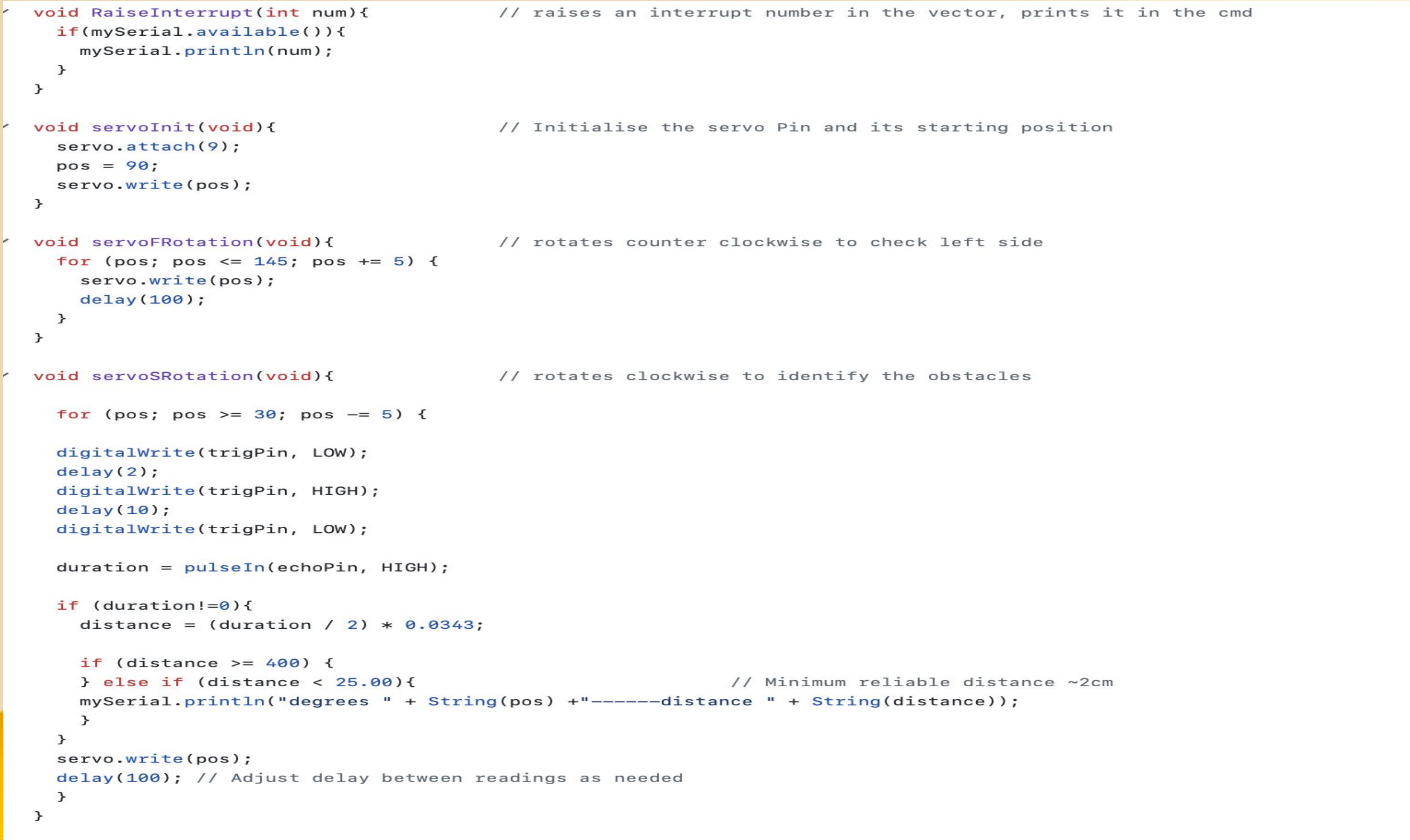


```
void loop() {                                            // put your main code here
  //if(PINTEST()){
  if (mySerial.available())
  {
    if(Travel && count == 10){
    mySerial.println("Scan Start");
    servoInit();
    servoFRotation();
    servoSRotation();
    servoReturn();
    Travel = false;
    mySerial.println("Scan end");
    }

    /*if(IsNear()){                  // sets Travel and Back to opposite valuse t
    Travel = true;
    Back = false;
    }*/

    BluetoothData=mySerial.read();
    Serial.println(BluetoothData);

    switch (BluetoothData){
      case 'F':
        goForward();
        set_Motorspeed(speedMotor,speedMotor);
        Travel = true;
        count = 0;
        break;
      case 'B':
        goBack();
        set_Motorspeed(speedMotor,speedMotor);
        Travel = true;
        count = 0;
                              // the rover backed enough from the obstacle, the U
        break;
      case 'L':
        goLeft();
        set_Motorspeed(speedMotor,speedMotor);
        Travel = true;
        count = 0;
        break;
      case 'R':
        goRight();
        set_Motorspeed(speedMotor,speedMotor);
        Travel = true;
        count = 0;
        break;
```



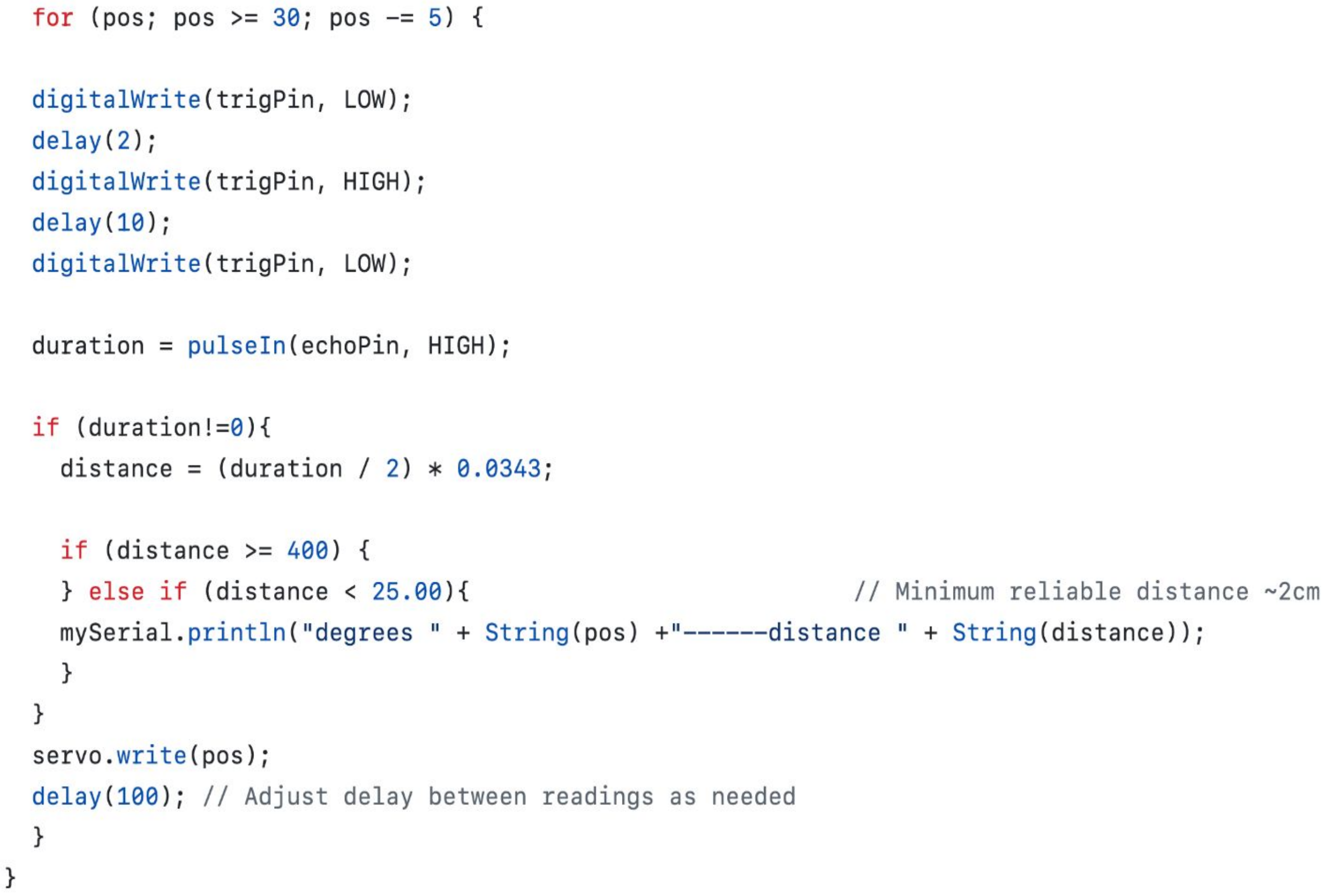
```
  case 'R':
   goRight();
   set_Motorspeed(speedMotor,speedMotor);
   Travel = true;
   count = 0;
   break;
  case 'S':
   count ++;
   Stop();
   break;
  case 'P':
   digitalWrite(LED_PIN,HIGH);
   if(temp==0){
    mySerial.println("LED ON");
    temp=1;
   }
   break;
  case 'W':
   digitalWrite(LED_PIN,LOW);
   if(temp!=0){
    mySerial.println("LED OFF");
    temp=0;
   }
   break;
  default:
   Stop();
   break;
 }
 //delay(20);                          // prepare for next data ...
}
```

Part of Ultrasonic Sensor



```
void RaiseInterrupt(int num){           // raises an interrupt number in the vector, prints it in the cmd
  if(mySerial.available()){
    mySerial.println(num);
  }
}

void servoInit(void){                   // Initialise the servo Pin and its starting position
  servo.attach(9);
  pos = 90;
  servo.write(pos);
}

void servoFRotation(void){              // rotates counter clockwise to check left side
  for (pos; pos <= 145; pos += 5) {
    servo.write(pos);
    delay(100);
  }
}

void servoSRotation(void){              // rotates clockwise to identify the obstacles

  for (pos; pos >= 30; pos -= 5) {

  digitalWrite(trigPin, LOW);
  delay(2);
  digitalWrite(trigPin, HIGH);
  delay(10);
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH);

  if (duration!=0){
    distance = (duration / 2) * 0.0343;

    if (distance >= 400) {
    } else if (distance < 25.00){                           // Minimum reliable distance ~2cm
    mySerial.println("degrees " + String(pos) +"------distance " + String(distance));
    }
  }
  servo.write(pos);
  delay(100); // Adjust delay between readings as needed
  }
}
```

Message Passing



```
  for (pos; pos >= 30; pos -= 5) {

  digitalWrite(trigPin, LOW);
  delay(2);
  digitalWrite(trigPin, HIGH);
  delay(10);
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH);

  if (duration!=0){
    distance = (duration / 2) * 0.0343;

    if (distance >= 400) {
    } else if (distance < 25.00){                           // Minimum reliable distance ~2cm
    mySerial.println("degrees " + String(pos) +"------distance " + String(distance));
    }
  }
  servo.write(pos);
  delay(100); // Adjust delay between readings as needed
  }
}
```