Group 1: Buenaventura, Delos Reyes, Louis

M3 Project – Final Documentation 08/03/2023

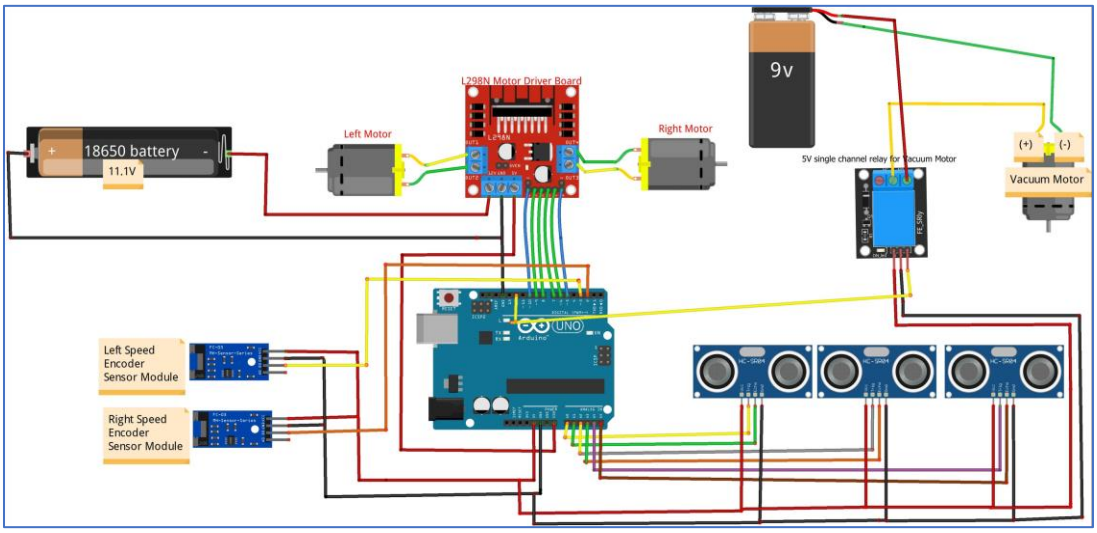

Figure 1. Schematic Diagram

Vacuum Bot Arduino IDE Code

```
// vacuum bot with ultrasonic sensor (events)
// [powerup] -->(A) --> [forward-all]--->(object_detect)-->[stop]--
>[optional:reverse all]----->[stop/wait]--->[reverse-left]--->(A)

///calibration headers
#define int_speed 100                              //adjust to slowdown vacuum bot
during [reverse left event]
#define int_speed_forward 70                       //adjust to slowdown vacuum bot
during [forward all event]
long frwd2rvrsleft_delay = 2000; //millisecond     //adjust the delay from
[forward all] to [reverse left event] when obstacle detected
bool withReverse = false;                          //if you want an additional
[reverse all event] after an obstacle been detected before to [reverse left
event]
#define Reverse_time 200 //millisecond            //adjust the [reverse all
event] if enable (withReverse=true)
```

```
bool useTwoRLUSensor = false;                       //set to "true" if you want to
use the Right and Left USensor as well for multiple sensor detection


// //ultrasonic
// #include "PID_v1.h"
// // PID tuning parameters (adjust these to fine-tune the controller)
// double Kp = 1.0;  // Proportional gain
// double Ki = 0.1;  // Integral gain
// double Kd = 0.1;  // Derivative gain

// // Define PID objects
// double setpoint = 10.0;     // Setpoint distance (adjust as needed)
// double input, output;
// PID myPID(&input, &output, &setpoint, Kp, Ki, Kd, DIRECT);
// const int maxSpeed = 255;    // Maximum motor speed
// const int minSpeed = 0;      // Minimum motor speed (0 to stop the motor)


#define forward false
#define reverse true
#define on true
#define off false
unsigned long spMillis1 = 0;
long sinterval1 = 2000;
volatile unsigned int sready1 = 0;

byte dataArray[100];
int fnum;
int snum;
int tnum;
int decimalValue;
int us_mode = 0;

// defines pins numbers
const int trigPin = A0;  //right
const int echoPin = A1;
const int trigPin2 = A2; //center
const int echoPin2 = A3;
const int trigPin3 = A4; //left
const int echoPin3 = A5;

// defines variables
long duration1;
int distance1;
```

```
long duration2;
int distance2;
long duration3;
int distance3;
int tachoval1;
int tachoval2;
// Motor A connections
int enA = 10;
int in1 = 9;
int in2 = 8;
// Motor B connections
int enB = 5;
int in3 = 7;
int in4 = 6;

const int sspin1 = 2;
volatile unsigned long counter1 = 0;

const int sspin2 = 3;
volatile unsigned long counter2 = 0;


void Right_Motor(bool status, int speed, bool direction){
  if(status){
    digitalWrite(in1, LOW);
    digitalWrite(in2, HIGH);
  }
  else {
    digitalWrite(in1, LOW);
    digitalWrite(in2, LOW);
  }
  analogWrite(enA, speed);
  if(direction){
    digitalWrite(in1, LOW);
    digitalWrite(in2, HIGH);
  }
  else{
    digitalWrite(in1, HIGH);
    digitalWrite(in2, LOW);
  }
}

void Left_Motor(bool status, int speed, bool direction){
  if(status){
    digitalWrite(in3, LOW);
```

```
    digitalWrite(in4, HIGH);
  }
  else{
    digitalWrite(in3, LOW);
    digitalWrite(in4, LOW);
  }
  analogWrite(enB, speed);
  if(direction){
    digitalWrite(in3, LOW);
    digitalWrite(in4, HIGH);
  }
  else{
    digitalWrite(in3, HIGH);
    digitalWrite(in4, LOW);
  }
}


void ReverseOn_All() {
  Right_Motor(true,int_speed,reverse);
  Left_Motor(true,int_speed,reverse);
}

void ReverseOff_All() {
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);

  // Right_Motor(false,127,reverse);
  // Left_Motor(false,127,reverse);
}

void ForwardOn_All() {
  Right_Motor(true,int_speed_forward,forward);
  Left_Motor(true,int_speed_forward,forward);
}

void ForwardOff_All() {

  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
```

```
  // Right_Motor(false,127,forward);
  // Left_Motor(false,127,forward);
}


void ReverseOn_Left(){
  Right_Motor(false,int_speed,forward);
  Left_Motor(true,int_speed,reverse);
}

void ReverseOn_Right(){
  Right_Motor(false,int_speed,reverse);
  Left_Motor(true,int_speed,forward);
}

void ReverseOn_Left_StopRight(){
  //stop right
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  //reverse left
  for (int i = int_speed; i >= 0; --i){
    Left_Motor(true,i,reverse);
    delay(20);
  }

}

// Interrupt service routine (ISR) for counting pulses
void pulseCount1() {
  counter1++;
}

// Interrupt service routine (ISR) for counting pulses
void pulseCount2() {
  counter2++;
}


void setup()
{

  Serial.begin(9600); // Starts the serial communication

//  // Set PID tuning parameters (optional)
//   myPID.SetMode(AUTOMATIC);
```

```
//   myPID.SetOutputLimits(minSpeed, maxSpeed);

//   // Update PID input and compute the control signal (motor speed)
//  input = distance;
//  myPID.Compute();
//  // Set the motor speed
//  int motorSpeed = (int)output;

  //Set all the motor control pins to outputs
  pinMode(sspin1, INPUT_PULLUP); // Set the IR sen2sor pin as input with internal
pull-up resistor
  attachInterrupt(digitalPinToInterrupt(sspin1), pulseCount1, RISING); // Attach
interrupt to D2 (interrupt 0)
  pinMode(sspin2, INPUT_PULLUP); // Set the IR sen2sor pin as input with internal
pull-up resistor
  attachInterrupt(digitalPinToInterrupt(sspin2), pulseCount2, RISING); // Attach
interrupt to D3 (interrupt 0)

  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
  pinMode(trigPin2, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin2, INPUT); // Sets the echoPin as an Input
  pinMode(trigPin3, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin3, INPUT); // Sets the echoPin as an Input

  // Set all the motor control pins to outputs
  pinMode(enA, OUTPUT);
  pinMode(enB, OUTPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);

  // Turn off motors - Initial state
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
}


void loop()
{

  unsigned long scMillis1 = millis();
```

```
  if (scMillis1 - spMillis1 >= sinterval1)
  {
    sready1 = 1;
    spMillis1 = scMillis1;
  }

  // if (Serial.available() > 0)
  // {
  //    int index = 0;
  //    delay(10); // let the buffer fill up
  //    int numChar = Serial.available();

  //    if (numChar>99) {
  //       numChar=99;
  //    }

  //    while (numChar--) {
  //       dataArray[index++] = Serial.read();
  //    }
  //     //Parsing here
  //     //led 1
  //     if((dataArray[0] == 'a'))
  //     {
  //       if(dataArray[1] >= 48){
  //         if(dataArray[1] >= 48 && dataArray[2] >= 48) {
  //           if(dataArray[1] >= 48 && dataArray[2] >= 48 && dataArray[3] >= 48)
{
  //             fnum = dataArray[1] - 48;
  //             snum = dataArray[2] - 48;
  //             tnum = dataArray[3] - 48;
  //             decimalValue = fnum * 100 + snum * 10 + tnum;
  //           }
  //           else {
  //             fnum = dataArray[1] - 48;
  //             snum = dataArray[2] - 48;
  //             decimalValue =  fnum * 10 + snum;
  //           }
  //         }
  //         else{
  //           fnum = dataArray[1] - 48;
  //           decimalValue = fnum;
  //         }
  //       }
  //         //Serial.println(decimalValue,DEC);
  //       for(unsigned int x = 0; x < 100; x++) dataArray[x] = 0;
```

```
//       Serial.flush();
//     }

// }

if(useTwoRLUSensor){
    // Clears the trigPin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration1 = pulseIn(echoPin, HIGH);
  // Calculating the distance
  distance1 = duration1 * 0.034 / 2;
  // Prints the distance on the Serial Monitor
  //Serial.print("Distance: ");
  //Serial.println(distance);
}



// Clears the trigPin
digitalWrite(trigPin2, LOW);
delayMicroseconds(2);
// Sets the trigPin on HIGH state for 10 micro seconds
digitalWrite(trigPin2, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin2, LOW);
// Reads the echoPin, returns the sound wave travel time in microseconds
duration2 = pulseIn(echoPin2, HIGH);
// Calculating the distance
distance2 = duration2 * 0.034 / 2;
// Prints the distance on the Serial Monitor
Serial.print("Distance2: ");
Serial.println(distance2);

if(useTwoRLUSensor){
  // Clears the trigPin
  digitalWrite(trigPin3, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin3, HIGH);
```

```
    delayMicroseconds(10);
    digitalWrite(trigPin3, LOW);
    // Reads the echoPin, returns the sound wave travel time in microseconds
    duration3 = pulseIn(echoPin3, HIGH, 1000000);
    // Calculating the distance
    distance3 = duration3 * 0.034 / 2;
    // Prints the distance on the Serial Monitor
    //Serial.print("Distance3: ");
    //Serial.println(distance3);
  }

  // static uint32_t previousMillis1;
  // if (millis() - previousMillis1 >= 1000) {
  //     tachoval1 = counter1;
  //     counter1 = 0;
  // }

  // static uint32_t previousMillis2;
  // if (millis() - previousMillis2 >= 1000) {
  //     tachoval2 = counter2;
  //     counter2 = 0;
  // }


//  Serial.print("U3:");
//  Serial.print(distance3);
//  Serial.print("|");
//  Serial.print("U2:");
//  Serial.print(distance2);
//  Serial.print("|");
//  Serial.print("U1:");
//  Serial.print(distance1);
//  Serial.print("|");
//  Serial.print("T1:");
//  Serial.print(tachoval1);
//  Serial.print("|");
//  Serial.print("T2:");
//  Serial.println(tachoval2);

  if(useTwoRLUSensor){
    if(distance1 <= 120) { //no detect
      us_mode = 0;
      Serial.println(us_mode);
      ForwardOn_All();
      //ReverseOff_All();
```

```
      //ReverseOn_Left();
      //ReverseOn_Right();
      //Serial.println("on");
    }
    if(distance1 >= 1000) {
      if(us_mode == 0) {
        us_mode = 1;
        spMillis1 = scMillis1;
      }
    }
     if(distance3 <= 120) { //no detect
      us_mode = 0;
      Serial.println(us_mode);
      ForwardOn_All();
      //ReverseOff_All();
      //ReverseOn_Left();
      //ReverseOn_Right();
      //Serial.println("on");
    }
    if(distance3 >= 1000) {
      if(us_mode == 0) {
        us_mode = 1;
        spMillis1 = scMillis1;
      }
    }
  }



 decimalValue = distance2;
//  Serial.println(duration2);
  if(decimalValue <= 120) { //no detect
    us_mode = 0;
    Serial.println(us_mode);
    ForwardOn_All();
    //ReverseOff_All();
    //ReverseOn_Left();
    //ReverseOn_Right();
    //Serial.println("on");
  }
  if(decimalValue >= 1000) {
    if(us_mode == 0) {
      us_mode = 1;
      spMillis1 = scMillis1;
    }
```

```
  }

  if(us_mode == 1) {
    if(sready1 == 1) {
      Serial.println("cs: obstacle=1 ");
      sready1 = 0;
      sinterval1 = frwd2rvrsleft_delay;
      spMillis1 = scMillis1;
      us_mode = 2;
      ForwardOff_All();
    }
  }

   if(us_mode == 2) {
    if(sready1 == 1) {
      Serial.println("cs: obstacle=1 timeout ");
      sready1 = 0;
      sinterval1 = frwd2rvrsleft_delay;
      spMillis1 = scMillis1;
      us_mode = 3;
      //reverse motor
      if(withReverse){
        ReverseOn_All();
        delay(Reverse_time);
      }
      //turn -left
      ReverseOn_Left_StopRight();
    }
  }


}
```