Autonomous Mobile Robot

Mini-Project

By

Jogesh S Nanda [P2.RAU16007]

Sreelekshmi P [P2.RAU16012]

Table of contents

Introduction

A robot which can move from one point to other is called a mobile robot. Due to advancement in technology and industrial need to adapt to new changes in manufacturing methods and logistics robots are in huge demand. Mobile robots are widely used in logistics and material handling to cater faster and efficient movement of materials. These mobile robots which work autonomously deliver the product from one location to other. Algorithm from higher hierarchy transmits only the destination coordinates to these mobile robots.

The mini-project done here is to get a hands on experience in building an autonomous mobile robot which can suit this demand. Differential drive robot is designed, build and tested in real world condition into which we give the position (x, y) coordinates to reach the destination avoiding unperceived obstacle.

Design

Aim/ problem statement:

Drive the mobile robot to a specific goal location given by (X, Y) co-ordinates through Bluetooth. Robot has to move to this specific goal location avoiding obstacles.

Approach:

Goal location is reached by moving the robot in its axis at each time. Robot is first moved along y axis then its turned 90 degree left or right and then moved through x axis.

Hardware Design:

Metal robot chassis was selected for better stability and control from fitting a high torque motor. Chassis was selected such that it has enough positions to mount the motor and other components such as driver or Arduino. ***Normally available wheel*** was purchased from the market and fitted with castor wheel.

Speed required (max speed of the robot) = 30 km/hr = 8.33 m/s

Radius of the wheel = 5 cm

So, RPM of motor required (min) = $velocity\ in\ m/s \,/\, radius\ of\ wheel$

= $\frac{8.33\ x\ 60}{5}$ = 100 rpm

DC motor of 100 rpm is selected for our purpose due to its high performance, easiness and reliability in controlling the speed and direction. Other motors such as induction, stepper offers lower torque to current ratio at low speeds. Moreover, DC motor driver is cheaper than other motors in this price range.

Motor driver we have chosen is ***L298N*** because it can drive upto 2 A per motor and provide a 5 V regulated output to drive the Arduino.

Other items such as ***Bluetooth module, ultrasonic sensor*** is purchased based on availability. ***Arduino Uno*** is used here as microcontroller because we require only 10 Digital I/O pins and 2 PWM and Uno is the cheapest Arduino which has 14 Digital I/O with 4 PWM.

Software Design:

Detailed flow chart to Programme is shown next page. Mathematical modelling used in this project are

Ultrasonic sensor – from sensor we get duration or time delay of sound wave to and fro in milliseconds

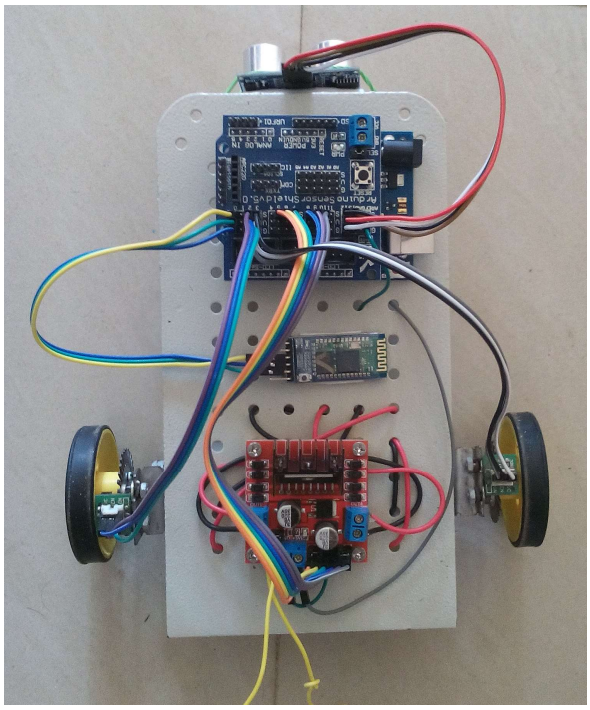
Distance = 0.034 x duration (unit – meter)

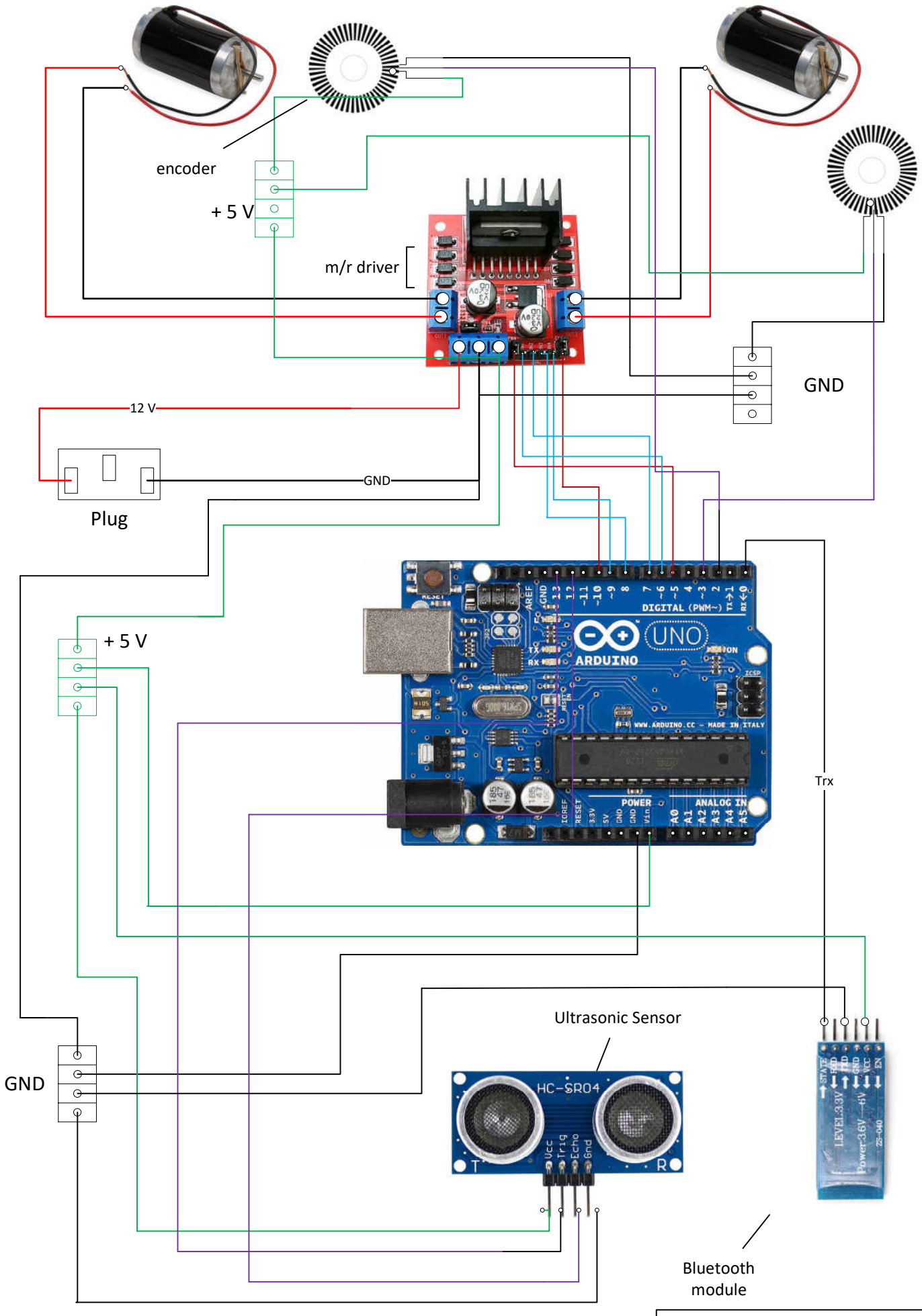
Encoder to distance conversion – radius of wheel is 2.5 cm. So, distance travelled by a rotation is 15.7 cm (2.pi.radius) and 32 encoder ticks are marked as one rotation

Distance = 15.7 x encoder value / 32 (unit – cm)

Circuit Diagram

Circuit diagram of the mobile robot is given in next page and completed wired robot is shown here.

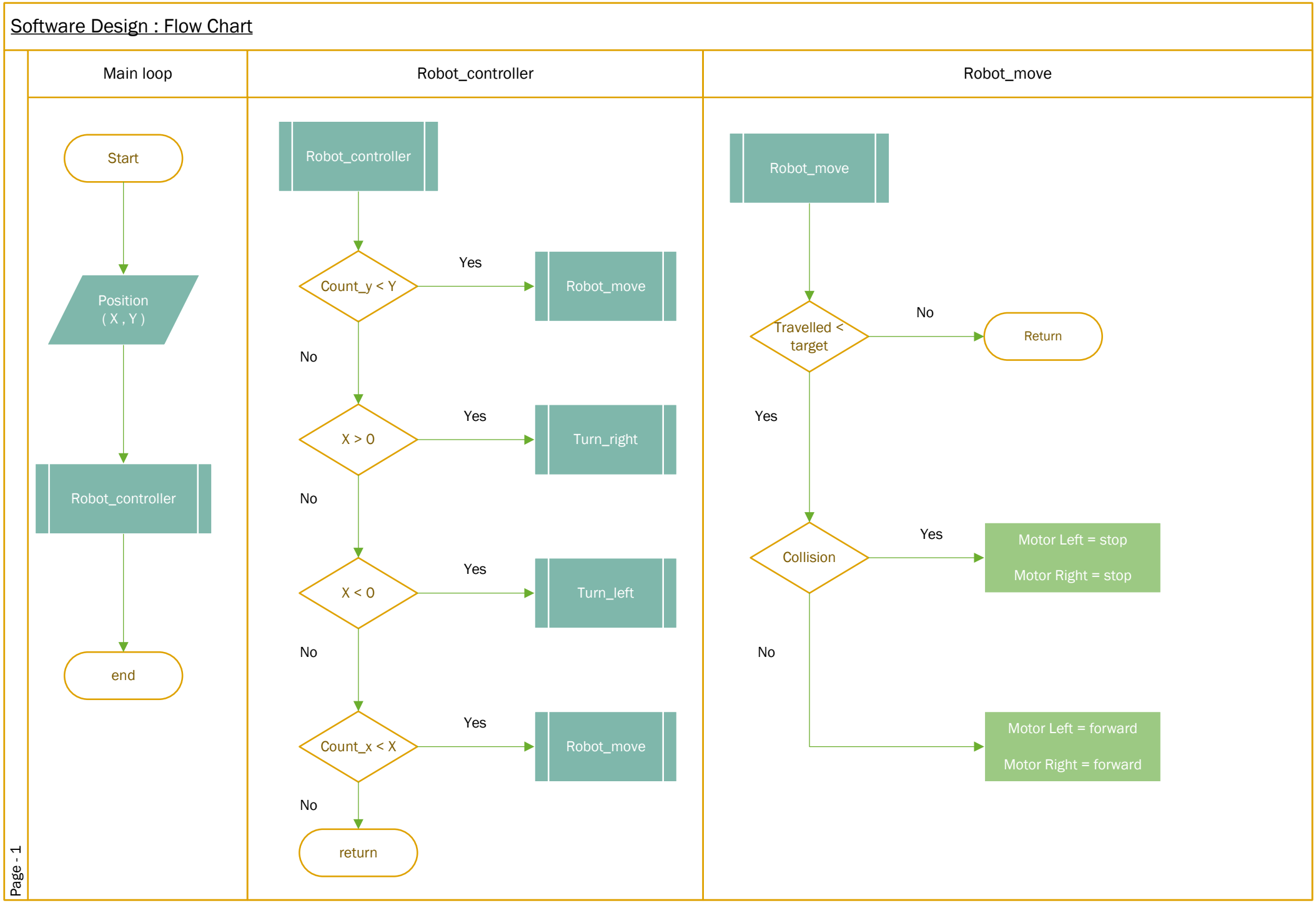




Circuit Diagram : Mobile Robot

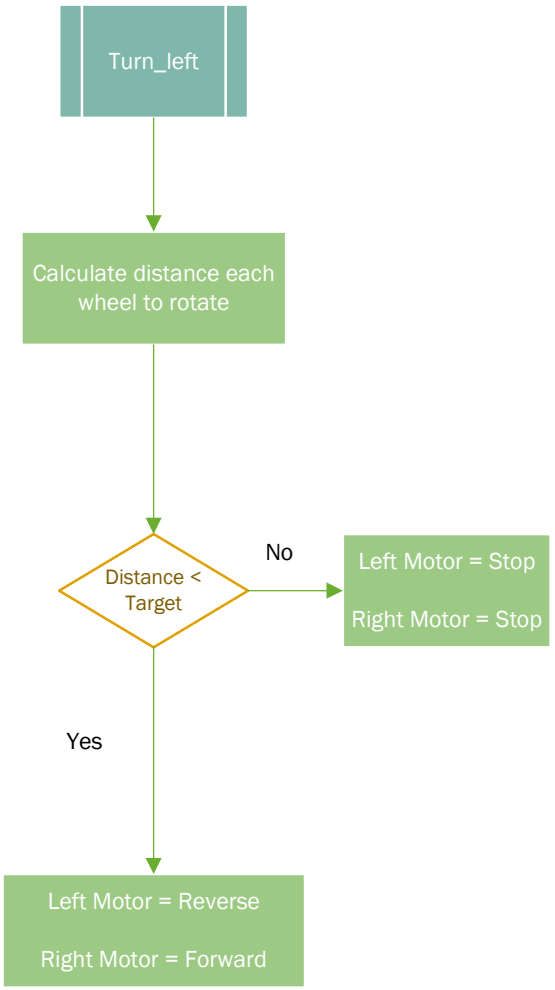
Jogesh S Nanda and Sreelekshmi P

Software Design : Flow Chart

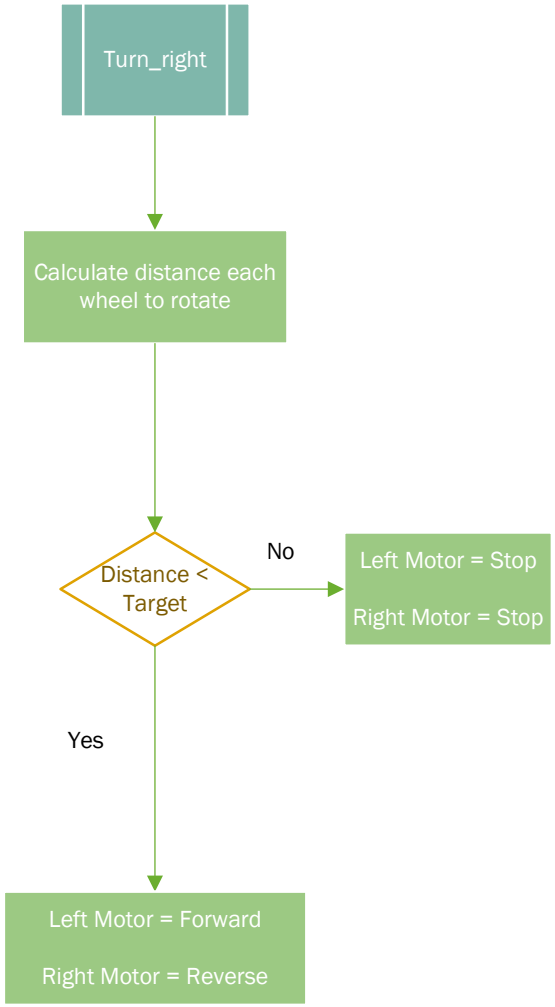


Software Design : Flow Chart

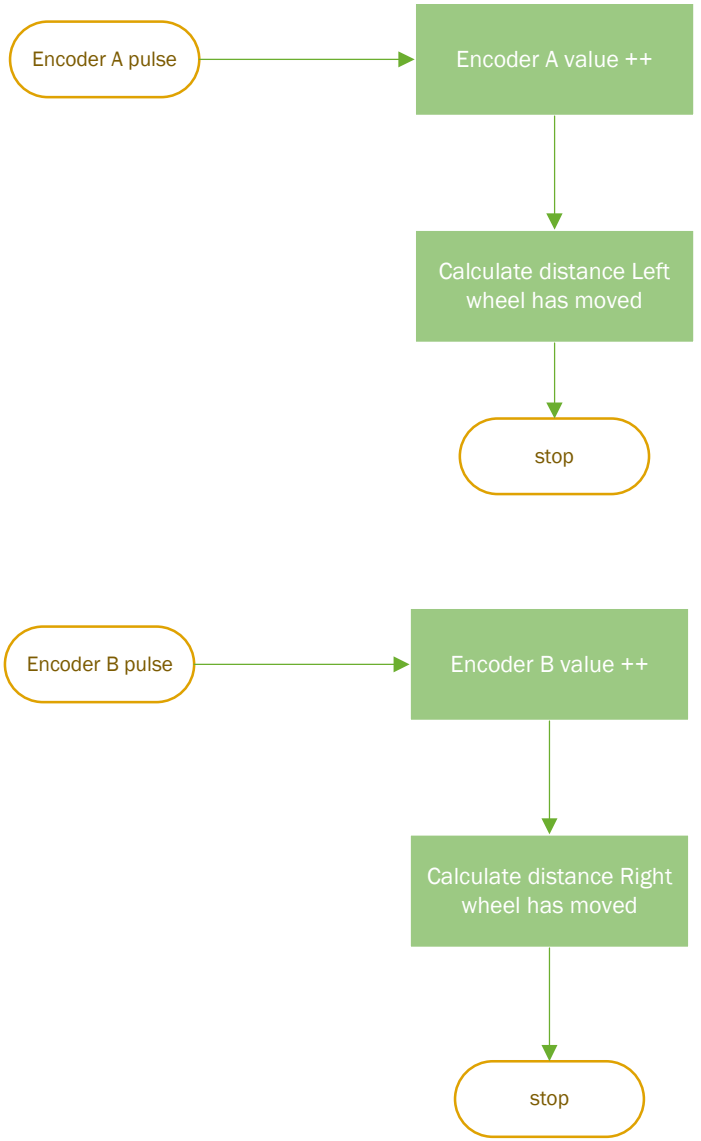
Turn_left



Turn_right



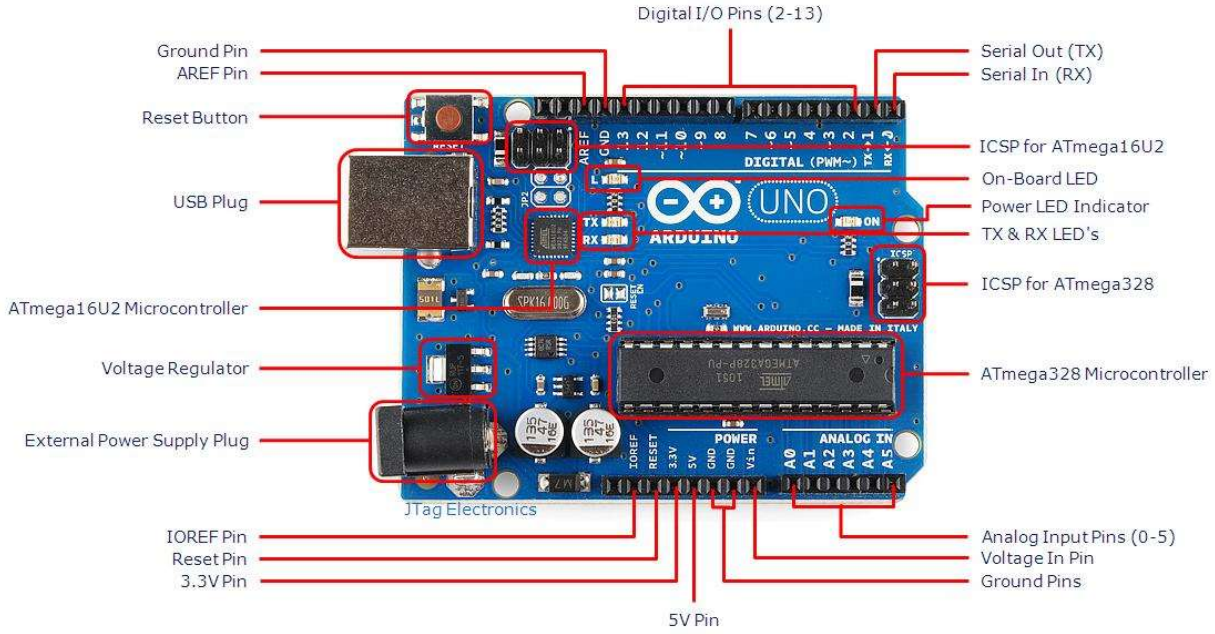
Interrupt Function



Components

Arduino Uno:

Arduino Uno is the cheapest development board based on AtMega328P microcontroller designed and developed by Arduino, Italy. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button.



Technical Specification

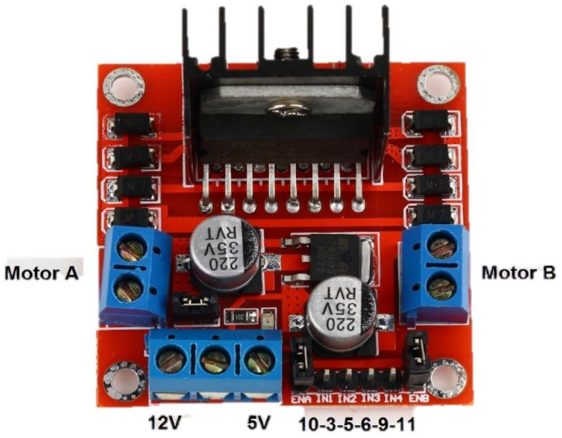
Microcontroller	AtMega328P
Operating Voltage	5 V
Digital I/o pins	14
PWM pins	6 marked as D0 to D13
Analog input Pins	6 marked as A0 to A5
DC current per I/O pin	20 mA
Flash memory	32 KB
Clock speed	16 MHz
Weight	25 g

Motor Driver:

Motor Driver we used this project is L298N driver. It's an H- bridge driver which can drive 2 DC motor or a single stepper motor up to 2A per H channel. This driver can control speed and direction of a motor by varying the setting in the control pin. Also within this module is a voltage regulator which can provide 5V output if needed depending on the jumper setting.

The driver can handle up to 35V input and if we are using greater than 12V the regulator IC jumper should be removed and in this case we cannot take the 5V regulated output.

Technical Specification:

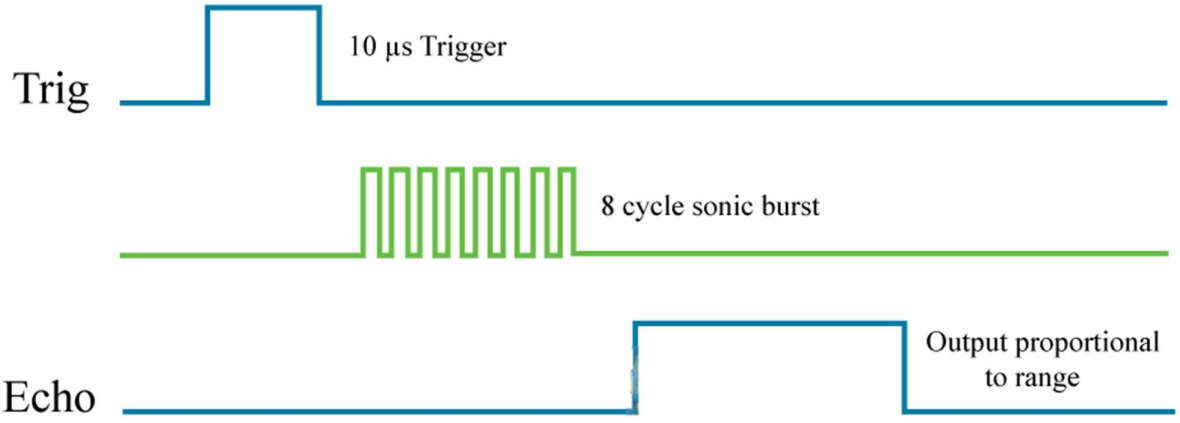


Motor A			
IN 1	***IN 2***	***ENA***	
High	Low	PWM	Motor A in forward direction
Low	High	PWM	Motor A in reverse direction
High	High		Stop
Low	Low		Stop
Motor B			
IN 2	***IN 3***	***ENB***	
High	Low	PWM	Motor B in forward direction
Low	High	PWM	Motor B in reverse direction

High	High		Stop
Low	Low		Stop

Ultrasonic Sensor:

Ultrasonic Sensor sends out a high frequency sound wave which gets deflected at a surface and returns back. Time taken for this process is calculated and we use this parameter to calculate the distance of the object from obstacle. The ultrasonic sensor used here is HC-SR04 has a range of 4 m. It gives out 8 cycles of echo when triggered and return signal is transmitted as echo high. Logic sequence of sensor in time domain is shown below.



Technical specification:

Working voltage	5 V
Current	15 mA
Range (min)	2 cm
Range (max)	400 cm
Angle of detection	15 degree
Ranging accuracy	3 mm
Working Frequency (sound waves)	40 KHz

For the proper working the trigger signal should be more than 10 µs, then the module automatically sends 40 KHz sound waves.

Time taken for sound waves to travel in air = 340 m/s

Time taken for return of echo signal = T ms (milliseconds)

Distance = $\frac{340\ x\ T}{2\ x\ 1000}$ = 0.034 x T/2

Motor:

Motor is a machine which converts the electric energy to rotational energy which can be used to dive the robot forward. Motor which we have used here is DC motor of 100 rpm.

Technical specification:

Operating Voltage	12 V
Rated speed	100 rpm from the output shaft
Torque	8 Kgf

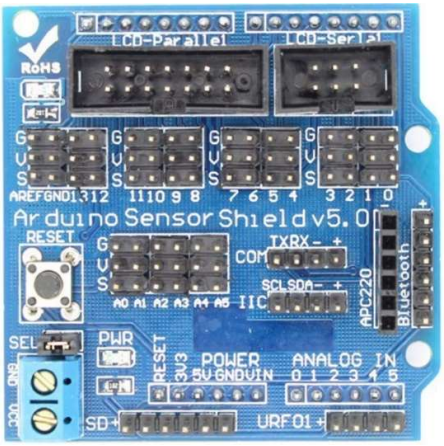
Encoder:

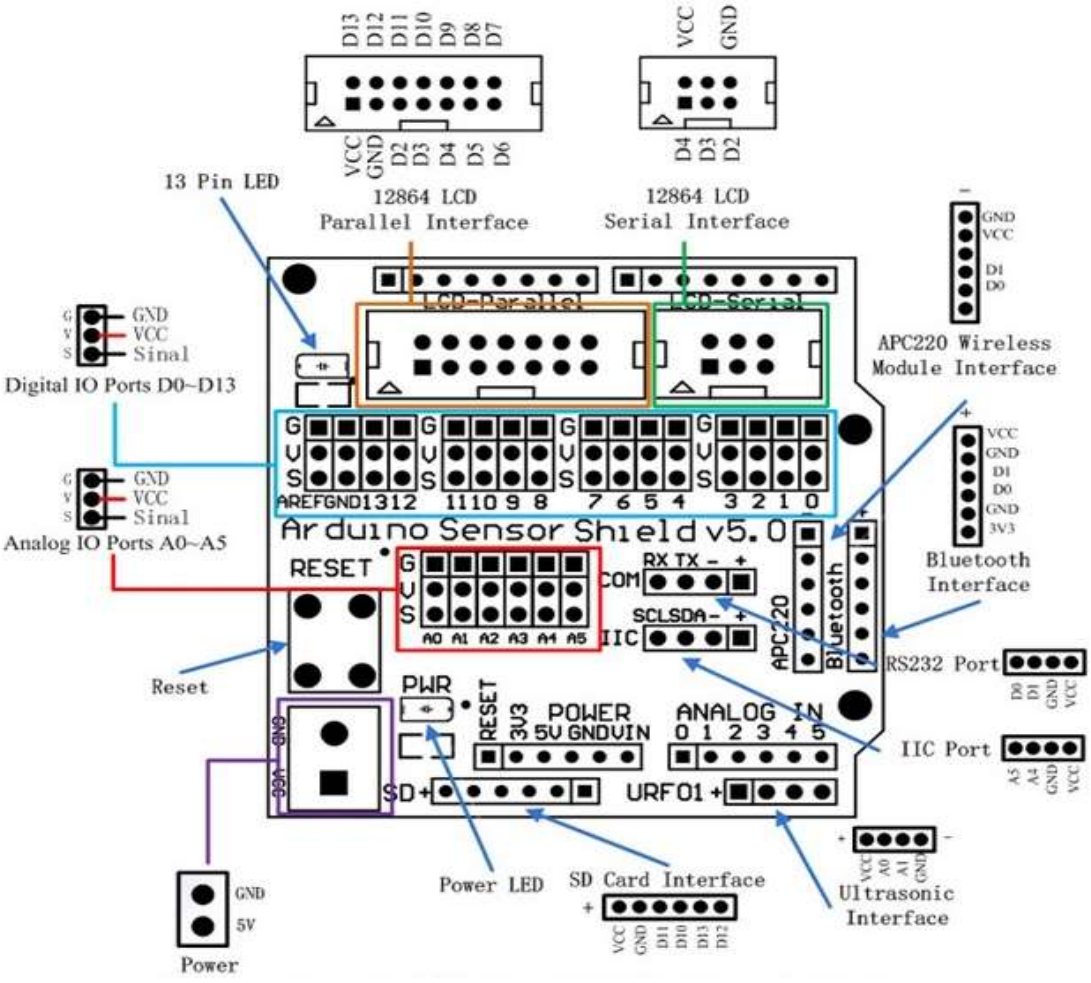
Encoder are connected to the wheels of the robot to get the information about how much each wheel has rotated in real world. Encoder used here is optical encoder which is mounted in a disk connected to the shaft of motor. Encoder gives high output when an obstacle is placed in between the transmission else low output.

The encoder disk which we have connected has 32 teeth so 32 pulses constitute a full rotation of wheel. Encoder is incremental type so we have to convert it to absolute value by dividing it with number of pulses per revolution.

Sensor Shield:

Sensor shield allows easy connection of sensors and in – out connection from Arduino. It's available in different versions for Arduino. Sensor shield we used here is Arduif Sensor Shield v5.0. It allowed us to take more than one Vcc and GND signal for each of our requirement.





Software Programme

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//          Global Variables Declaration
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// declaring the global variable for converting incremental encoder to absolute encoder
        volatile long encoder_A_pos = 0;          // incremental encoder value for Left wheel
        volatile long encoder_B_pos = 0;          // incremental encoder value for Right wheel

// distance travelled by the motor wheel
         float distance_motor_A = 0;      // distance travelled by left wheel
         float distance_motor_B = 0;       // distance travelled by right wheel
// Bluetooth
        char bluetooth ;
// Current Position of Robot
        int count_x = 0;
        int count_y = 0;

// job = 0 ===> not completed
// job = 1 ===> moved forward
// job = 2 ===> taken a turn
        int job = 0;

// stage = 1 ===> has to move in Y co-ordinate
// stage = 2 ===> has to take left or right 90 degree turn
// stage = 3 ===> has to move in X co-ordinate
        int stage = 1;

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//          PORT initialization
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// Ultrasonic Sensor
        int trig = 13;              // Trigger switch to send the echo sound.
        int echo = 12;              // For the detection of return echo.
// motor LEFT wheel
```

```
        int enA = 5;      // pwm pin to control the motor speed
        int in1 = 6;      // direction control pin
        int in2 = 7;      // direction control pin
// motor RIGHT wheel
        int in3 = 8;      // direction control pin
        int in4 = 9;      // direction control pin
        int enB = 10;     // pwm pin to control the motor speed
// encoder pulse from motor
        int encoder_A = 2;        // encoder from left wheel
        int encoder_B = 3;        // encoder from right wheel

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//          Motor control
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// Function: motor_A
Void motor_A (int speed, word direction) {
        // to drive the motor in forward direction
        if (direction == "forward") {
                digitalWrite (in1, HIGH);
                digitalWrite (in2, LOW);
        }
        // to drive the motor in reverse direction
        else if ( direction == "backward") {
                digitalWrite (in1, LOW);
                digitalWrite (in2, HIGH);
        }
        // STOP the motor
        else if (direction == "stop") {
                 digitalWrite (in1, HIGH);
                digitalWrite (in2, HIGH);
          }
        // to drive the motor at speed from value 0 to 255
        analogWrite (enA, speed);
}
```

```
// Function: motor_B
Void motor_B (int speed, word direction) {
        // to drive the motor in forward direction
         if (direction == "forward") {
                  digitalWrite (in3, HIGH);
                   digitalWrite (in4, LOW);
          }
        // to drive the motor in reverse direction
        else if (direction == "backward") {
                  digitalWrite(in3, LOW);
                  digitalWrite(in4, HIGH);
         }
        // to STOP the motor
        else if (direction == "stop") {
                  digitalWrite(in3, HIGH);
                  digitalWrite(in4, HIGH);
        }
         // to drive the motor at speed from value 0 to 255
        analogWrite(enB, speed);
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//        encoder
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// motor_A_encoder ---> from the interrupt function increment Variable “encoder_A_pos " each time it's called
// motor_B_encoder ---> from the interrupt function increment Variable “encoder_B_pos " each time it's called
// distance_motor_A ---> Global Variable which gives the distance travelled by the Left Wheel
// distance_motor_B ---> Global Variable which gives the distance travelled by the Right Wheel
// distance_reset ---> reset the encoder counter

Void motor_A_encoder () {
        // encoder reading
        encoder_A_pos = encoder_A_pos + 1;

        // calculate the distance travelled by both the motors.   --> Unit: cm
```

```
        // radius of wheel ===> 2.5 cm
         // number of teeth ===> 32 nos
        // distance for 1 wheel rotation = 2 x pi x radius of wheel ---> in in this case is 16
        // distance travelled by wheel = distance travelled by 1 wheel rotation x total number of rotation
        // total number of rotation = encoder_pos / number of teeth (here 32)
        // distance_of_motor = ( encoder_pos / number of teeth ) x 2.pi x radius of wheel
         distance_motor_A = encoder_A_pos / 2 ;
}

Void motor_B_encoder () {
        // encoder reading
        encoder_B_pos = encoder_B_pos + 1;

        // calculate the distance travelled by both the motors.   --> Unit: cm
        // radius of wheel ===> 2.5 cm
        // number of teeth ===> 32 nos
        // distance for 1 wheel rotation = 2 x pi x radius of wheel ---> in in this case is 16
        // distance travelled by wheel = distance travelled by 1 wheel rotation x total number of rotation
        // total number of rotation = encoder_pos / number of teeth (here 32)
        // distance_of_motor = (encoder_pos / number of teeth ) x 2.pi x radius of wheel
        distance_motor_B = encoder_B_pos / 2 ;
}

Void distance_reset () {
        // reset the distance travelled by both the motors for new function.
        encoder_A_pos = 0;
         encoder_B_pos = 0;

        delay(2000);
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//          Ultrasonic Sensor
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// gives the distance from Obstacle in (cm)
```

```
float ultrasonic_sensor() {
 // declaration of local Variables
          unsigned long duration;    // time taken by the echo pulse
          float distance = 0;          // distance from Obstacle

 // Creating a Pulse for TRIG. (Min duration needed is 10 us).
          digitalWrite(trig, LOW);
          delay (10);
          digitalWrite(trig, HIGH);
          delay(10);
          digitalWrite(trig,LOW);

          duration = pulseIn (echo, HIGH);

 // Calculating Distance from Time
          distance = 0.034 * duration/2;

          return(distance);
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//          Robot Dynamics
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

// robot_move ---> move the robot at distance X [in cm]
// robot_turn_left ---> turn the robot at angle (anti-clockwise)
// robot_turn_right ---> turn the robot at an angle (clock-wise)
// robot_stop ---> emergency STOP

// Function: Robot_move
void robot_move (int target) {
          float travelled, collision_distance;

 // averaging the distance or Filtering
          travelled = (distance_motor_A + distance_motor_B) /2 ;
```

```
 // measuring collision distance from ultrasonic sensor.
        collision_distance = ultrasonic_sensor ();

 // Condition for moving the robot forward.
         if ( travelled < target) {

        // emergency stop if obstacle is found.
                if (collision_distance <20) {
                        robot_stop ();     // min. obstacle avoidance distance ---> 20 cm
                }
        // robot move forward if no obstacle is found.
                else {
                        motor_A (100, "forward");
                         motor_B (78, "forward");
                 }
        }
        else {
                motor_A (255, "stop");
                motor_B (255, "stop");
                // setting the Flag as done
                job = 1;
                delay(100);
        }
}

// Function: robot_turn_left
// input ===> angle to move (in degree)
void robot_turn_left(int angle) {
        float distance, target;

 // distance to turn is radius x Angle (in radian)
 // length/radius ===> 18.5 cm
 // angle(radian) ===> (angle/180 ) x 3.14
 // distance ===> angle(radian) x radius = angle x 0.314 = angle / 3
```

```
 // Calculating distance to rotate the wheel
        target = angle/9 - 0;
        distance = (distance_motor_A + distance_motor_B) / 2;

 // Left wheel Rotation
        if (distance < target) {
                motor_A(255, "backward");
         }
        else {
                 motor_A(255,"stop");
                 // setting the Flag as done.
                job = 2;
         }

 // Right Wheel Rotation
        if (distance < target) {
                 motor_B (255, "forward");
         }
         else {
                motor_B (255,"stop");
                // setting the Flag as done.
                job = 2;
        }
}

void robot_turn_right (int angle) {
        float distance, target;

 // distance to turn is radius x Angle (in radian)
 // length/radius ===> 18.5 cm
 // angle (radian) ===> (angle/180 ) x 3.14
 // distance ===> angle (radian) x radius = angle x 0.314 = angle / 3

 // Calculating distance to rotate the wheel
```

```
        target = angle/9 - 0;

         distance = (distance_motor_A + distance_motor_B) / 2;

 // Left wheel Rotation
        if (distance  < target) {
                motor_A (255, "forward");
         }
        else {
                motor_A (255,"stop");
                // setting the Flag as done.
                job = 2;
        }

 // Right Wheel Rotation
        if (distance < target) {
                motor_B (255, "backward");
         }
         else {
                motor_B (255,"stop");
                // setting the Flag as done.
                job = 2;
         }
}

// robot_stop ---> Emergency Stop
Void robot_stop () {
        motor_A (255, "stop");
        motor_B (255, "stop");

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//          Robot Control Autonomous
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
Void robot_controller (int x, int y) {
```

```
 // Stage: 1
// Move the robot in Y Co-ordinate
        If (stage == 1) {
                If (count_y <= y) {
                        robot_move(y);
                        count_y = robot_distance ();
                }

        // checking if stage 1 is complete
         if (job == 1) {
                distance_reset ();
                Stage = 2;
                delay (100);
         }
}

// stage: 2
// turn the robot towards appropriate Direction.
        If (stage == 2) {
        // Turn the robot towards + ve X Co-ordinate
                 If (x > 0) {
                        robot_turn_right (90);
                }
         // Turn the robot towards -ve X Co-ordinate
                Else if (x < 0) {
                        robot_turn_left (90);
                }

        if (job == 2) {
                distance_reset ();
                stage = 3;
                delay(100);
         }
 }
```

```
 // Stage: 3
 // Move the robot in X Co-ordinate.
          if (stage == 3) {
                    // Move robot forward
                    robot_move (abs(x));
                    count_x = robot_distance ();
          }
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//          Setup
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
Void setup () {
 // setting the control signals of Motor A OUTPUT
          pinMode (enA, OUTPUT);
          pinMode (in1, OUTPUT);
          pinMode (in2, OUTPUT);

 // setting the control signals of Motor B as OUTPUT
          pinMode (enB, OUTPUT);
          pinMode (in3, OUTPUT);
           pinMode (in4, OUTPUT);

 // setting the encoder pins an INPUT
           pinMode (encoder_A, INPUT);
           pinMode (encoder_B, INPUT);

 // setting Ultrasonic Sensor
          pinMode (trig, OUTPUT);
          pinMode (echo, INPUT);

 // Start the baud rate of serial plotting in Arduino
          Serial.begin (9600);
```

```
 // attaching the intrupt function for reading encoder values.
 // interupt 0 == pin 2 ---> set for rising edge detection ---> Motor A
 // interupt 1 == pin 3 ---> set for rising edge detection ---> Motor B
            attachInterrupt (0, motor_A_encoder, RISING);
            attachInterrupt (1, motor_B_encoder, RISING);

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//          Main Loop
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
Void loop () {

 // Checking the Availabity of Bluetooth Communication
        if (Serial.available () > 0) {
                bluetooth = Serial.read ();
        }

        robot_controller (60, 0);

}
```

Testing

Sl. No	***Item***	***Tested***	***Comment***
1	Arduino Uno	Ok	Digital I/o pins and analog pins tested using a multimeter
2	Ultrasonic Sensor	Ok	Min range – 2 cm to detect (tested) Max range – 4.1 m (tested) Detects only object in straight line
3	Bluetooth module	Ok	
4	L298N motor driver	Ok	Reported issue with different voltage at 2 motor terminal for same PWM. This error was calibrated using software correction
5	DC motor	Ok	Max motor rpm – 100 (tested when 12 V given directly) Motor reported high inertia for small changes so breaking was difficult.
6	Optical encoder	Ok	32 pulse per revolution of motor wheel (tested) but direction of rotation of wheel cannot be determined
7	Castor wheel	Ok	High friction, wear and tear causes errors during turning
8	Wheels	Ok	

Known Bugs:

1. Due to error in L298N motor driver we have to give different PWM to each motor to drive it in straight line (by trial and error method) [ref. “robot_move” function in Software section] but this correction leads to slight variation in obtaining accurate angle position.
2. To get accurate angle value motor has to drive in lower speeds because motor driver do not have a breaking system so it moves another 2 – 4 cm (tested) before coming to rest. This has to be taken into considerations when calculating distance both wheels to rotate to get a

particular angle. Here we have calibrated for getting accurate 90 degree so this leads to minor errors for other angles.

3. Since encoder does not give the value of direction any slip causes error in Odometer calculation which can leads to cumulative errors while working

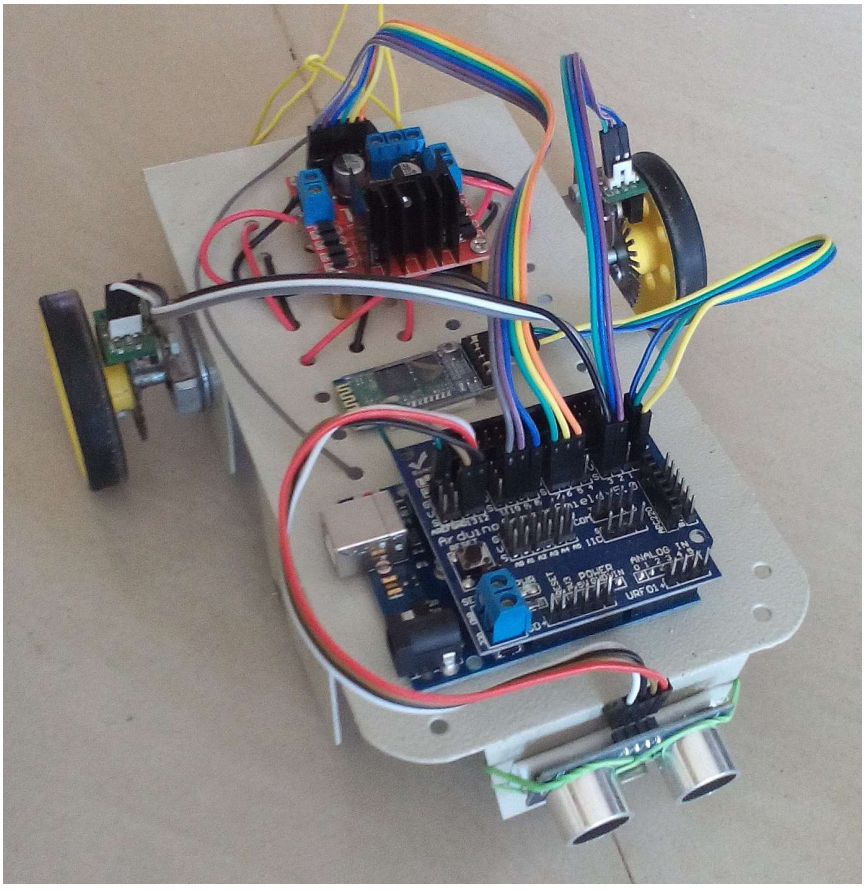
Bill of Materials

Sl. No	***Item***	***Unit Price***	***Qty.***	***Total***
1	Arduino Uno	455	1	455
2	Ultrasonic sensor HC-SR04	249	1	249
3	Bluetooth module	335	1	335
4	L298N motor driver module	260	1	260
5	Robot metal chassis	150	1	150
6	12 V DC motor 100 rpm	500	2	1000
7	Optical encoder	360	2	720
8	Castor wheel mini for small robot	28	1	28
9	Connector pins (male-male, female-female, and male-female)	220	1	220
10	Robot wheel 4 PCS set	125	1	125
	Total			3,542

Conclusion

An autonomous mobile robot was build using differential drive mechanism which was controlled by an Arduino UNO board. The robot was tested by giving a coordinate position and the robot reached the point within the desired accuracy.

We used commonly available parts to build the robot and overall cost builds to 3,542 INR. The parts and robot was tested at each stage of progress to get a continuity of coding and for easier debugging of errors. Entire robot movement algorithm was built on odometer calculation of each wheel of the robot because encoder reading was the most reliable feedback tested in our components. Also, coding was much easier in implementing odometer based calculation.



Reference

[1] DESIGN OF A DIFFERENTIAL DRIVE MOBILE ROBOT PLATFORM FOR USE IN CONSTRAINED ENVIRONMENTS by A.V. Chavan and Dr. J. L. Minase

[2] On Differential Drive Robot Odometry with Application to Path Planning Evangelos Papadopoulos and Michael Misailidis

[3] Adrift Motor shield manual

[4] Arduino.com