

## LESSON 7

### **Collision Detection**

In this lesson, you will program your Pi-Bot to prevent accidents.

Combining with what we have learned about the ultrasonic sensor, together using interrupt control and motion control, we shall now investigate obstacle avoidance.

Using the program shown in figure 7.1, your Pi-Bot will continue forward at a medium speed until an obstacle is detected within 10cm of the sensor. Upon detecting the object in front of the sensor, your Pi-Bot will reverse in a backward turn until the obstacle is no longer in front of the sensor plus 1 second after all obstacles have left the field of view.

#### Exercise 1

Modify the forward speed and the backward speed of your Pi-Bot. Try different speeds on the two wheels when traveling in reverse.

#### Exercise 2

Upon detecting an obstacle, have your Pi-Bot reverse in different directions.

For example, upon detecting the first obstacle, have your Pi-Bot reverse while turning to the right. The next time it detects an obstacle, have your Pi-Bot turn to the left.

```

/*
Ultrasonic sensor controlling an led
*/

#define INTNUM 1 //interrupt pin 1 is digital pin 3 on the Arduino
#define PULSE 10 //microseconds
#define CYCLETIME 50 //milliseconds - this is the time to wait until the
sensor is asked to find
// the distance again. #times/second = 1000/CYCLETIME =
20 in this case
#define Close 10 // Distance in cm to turn on the Led => Led is on if
distance is less than 10 cm

int millisNow, millisPrev = 0;
int microsPrev;
const int TrigPin = 2; // trigger pin
const int EchoPin = 3; //interrupt-enabled pin
const int Led = 13; // led pin
const int In4 = 11; // In4
const int In3 = 10; // In3
const int In2 = 6; // In2
const int In1 = 5; // In1

boolean isHigh = false;
boolean dir;

void setup()
{
  Serial.begin (9600);
  pinMode(TrigPin, OUTPUT);
  pinMode(EchoPin, INPUT);
  pinMode(In1, OUTPUT);
  pinMode(In2, OUTPUT);
  pinMode(In3, OUTPUT);
  pinMode(In4, OUTPUT);

  attachInterrupt(INTNUM, getTime, CHANGE); // Calls getTime when an change
occurs on pin 3
}

void loop()
{
  int speed1, speed2;
  trigPulse();
  speed1 = 200;
  speed2 = 200;
  forward(speed1, speed2);
  if(dir == false)
  {
    speed1 = 100;
    speed2 = 200;
    reverse(speed1, speed2);
    delay(2000);
    dir=true;
  }
}

void trigPulse()
{
  if( (millisNow = millis()) - millisPrev >= CYCLETIME) //sufficient cycle
time
  {
    digitalWrite(TrigPin, HIGH);
    delayMicroseconds(PULSE);
    digitalWrite(TrigPin, LOW);
    millisPrev = millisNow; //reset clock
  }
  return;
}

```

```

void ReverseRobot(int dTime)
{
    int speed1, speed2;
    int delayTime;
    int distance;

    noInterrupts(); //lets not get interrupted while we are processing the
first interrupt
    distance = dTime/58; // distance in cm
    Serial.print(distance);
    Serial.println(" cm");
    if (distance < Close)
    {
        dir=false;
    }
    interrupts(); //turn interrupts back on
}

void getTime() //calling micros() here returns micros when function was
called. No increment during interrupt
{
    if(isHigh == false) // pin LOW->HIGH
    {
        microsPrev = micros();
        isHigh = true;
        return;
    }
    else //pin HIGH->LOW
    {
        ReverseRobot(micros() - microsPrev);
        isHigh = false;
        microsPrev = micros();
        return;
    }
    return;
}

void forward(int speed1, int speed2)
{
    analogWrite(In4, speed1);
    analogWrite(In3, 0);
    analogWrite(In2, 0);
    analogWrite(In1, speed2);
}

void reverse (int speed1, int speed2)
{
    analogWrite(In4, 0);
    analogWrite(In3, speed1);
    analogWrite(In2, speed2);
    analogWrite(In1, 0);
    return;
}

void stopNow()
{
    analogWrite(In4, 0);
    analogWrite(In3, 0);
    analogWrite(In2, 0);
    analogWrite(In1, 0);
    return;
}

```

*Figure 7.1: Program – obstacle1.ino*