A PRACTICAL ACTIVITY REPORT SUBMITTED FOR

ENGINEERING DESIGN-III (UTA011)

End-Semester Lab Evaluation

Submitted by:

Debangshu Mukherjee

Debarshi Ghosh

Deeksha Mani

Deepak Gupta

Submitted To:

Simarpreet Singh

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

TIET, Patiala

January-May 2019

Abstract

In this project of buggy we learned about the various challenges in buggy- bronze, silver and gold. We learned how we can move buggy with the codes. Also we learnt about the coding of Arduino Uno that involved the basic concepts of C++ and C Programming that we had studied last semester. Through this project we have also learnt about the usage of buggy in our daily lives and its various components that would enhance our knowledge of basic electronics.

Declaration

Our group members would like to thank Simarpreet Singh, our teacher who helped us at every moment of performing our experiments. Through his help and guidance we successfully completed all our experiments that we were being told to perform. Also we enhanced our knowledge regarding this subject.

Table of Contents

Experiment:1

Objective: To demonstrate the use of Ultrasonic sensor by integrating the line following RoboCar with obstacle avoidance capability

Hardware Used: Nvis 3302ARD RoboCar, Ultrasonic sensor HC-SR04

Software Used: Arduino IDE

Theory:

1. **pinMode()** - Configures the specified pin to behave either as an input or an output. Syntax of pinMode() function is:pinMode(pin, mode);

2. **digitalWrite()** - If the pin has been configured as an OUTPUT with pinMode(), its voltage will be set to the corresponding value: 5V (or 3.3V on 3.3V boards) for HIGH, 0V (ground) for LOW. If the pin is configured as an INPUT, digitalWrite() will enable (HIGH) or disable (LOW) the internal pullup on the input pin. Syntax:digitalWrite(pin, value);

3. **digitalRead()** - Reads the value from a specified digital pin, either HIGH or LOW. Syntax:digitalRead(pin, value);

4. **Trig Pin –** It is a trigger pin where we need to provide a trigger after which this sensor emits ultrasonic wave. pinMode(trigPin, OUTPUT);

5. **Echo Pin –** When ultrasonic waves emitted by the transmitter, hit some object then they are bounced back and are received by receiver at that moment echo pin goes high. pinMode(echoPin, INPUT);

In this we use the NewPing.h library to integrate the ultrasonic sensor with the RoboCar. We declare the maximum ping distance as 200cm. The trigger pin is 13 and echo pin is 12. The robocar communicates with the computer using the Serial library at 9600 baud rates.

An Ultrasonic Sensor is a device that can measure the distance to an object by using sound waves. It measures distance by sending out a sound wave at a specific frequency and listening for that sound wave to bounce back. By recording the elapsed time between the sound wave being generated and the sound wave bouncing back, it is possible to calculate the distance between the sonar sensor and the object.

Code:

```
int t1=A0; //IR sensor
int t2=A2; //IR sensor
int pin5=5; //Motor pin
int pin6=6; //Motor pin
int pin7=8; //Motor pin
int pin8=7; //Motor pin
```

```
//For Ultrasonic Sensor
#include <NewPing.h>
#define TRIGGER_PIN 13
#define ECHO_PIN 12
#define MAX_DISTANCE 200
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);

void setup() {
pinMode(pin5,OUTPUT);
pinMode(pin6,OUTPUT);
pinMode(pin7,OUTPUT);
pinMode(pin8,OUTPUT);
pinMode(t1,INPUT);
pinMode(t2,INPUT);
Serial.begin(9600);
}

void lineFollow(){
int r1=digitalRead(t1);
int r2=digitalRead(t2);
if(r1==LOW&&r2==LOW)
{
        digitalWrite(pin5,HIGH);
        digitalWrite(pin6,LOW);
        digitalWrite(pin7,HIGH);
        digitalWrite(pin8,LOW);

}
if(r1==HIGH&&r2==LOW)
{
        digitalWrite(pin5,LOW);
        digitalWrite(pin6,LOW);
        digitalWrite(pin7,HIGH);
        digitalWrite(pin8,LOW);

}
if(r1==LOW&&r2==HIGH)
{
        digitalWrite(pin5,HIGH);
        digitalWrite(pin6,LOW);
        digitalWrite(pin7,LOW);
        digitalWrite(pin8,LOW);
}
if(r1==HIGH&&r2==HIGH)
{
        digitalWrite(pin5,HIGH);
        digitalWrite(pin6,LOW);
        digitalWrite(pin7,HIGH);
        digitalWrite(pin8,LOW);
}
}

void stopBuggy(){
        digitalWrite(pin5,LOW);
        digitalWrite(pin6,LOW);
        digitalWrite(pin7,LOW);
        digitalWrite(pin8,LOW);
}

void loop(){
unsigned int dCM;
dCM = sonar.ping_cm();
pinMode(ECHO_PIN,OUTPUT);
digitalWrite(ECHO_PIN,LOW);
pinMode(ECHO_PIN,INPUT);

Serial.print("Ping: ");
Serial.println(dCM);
Serial.println("cm");
if((dCM<15) && (dCM>0))
{
        stopBuggy();
        delay(1000);
}
else
{
        lineFollow();
}
}
```

Result Analysis:

We attached the ultrasonic sensor to the buggy and moved the buggy. We tested the code of this sensor by placing obstacles in front of the robocar, and it worked correctly.

Experiment – 2

Objective – Write a program to read the pulse width of the gantry transmitter and trigger stop_buggy function by detecting individual gantry.

Hardware Used- Nvis 3302ARD RoboCar, PWM transmitter, PWM receiver

Software Used- Arduino IDE 1.8.4

Theory-

The Digital Input/output functions used in the program are:

1. **pinMode()** - Configures the specified pin to behave either as an input or an output. Syntax of pinMode() function is:pinMode(pin, mode);

2. **digitalWrite()** - If the pin has been configured as an OUTPUT with pinMode(), its voltage will be set to the corresponding value: 5V (or 3.3V on 3.3V boards) for HIGH, 0V (ground) for LOW. If the pin is configured as an INPUT, digitalWrite() will enable (HIGH) or disable (LOW) the internal pullup on the input pin. Syntax:digitalWrite(pin, value);

3. **digitalRead()** - Reads the value from a specified digital pin, either HIGH or LOW. Syntax:digitalRead(pin, value);

The Analog Input/Output functions used in Arduino are:

1. **analogRead()** - Reads the value from the specified analog pin. The Arduino board contains a 6 channel (8 channels on the Mini and Nano, 16 on the Mega), 10-bit analog to digital converter. This means that it will map input voltages between 0 and 5 volts into integer values between 0 and 1023. This yields a resolution between readings of: 5 volts / 1024 units or, .0049 volts (4.9 mV) per unit. Syntax:analogRead(pin);

2. **analogWrite()** - Writes an analog value (PWM wave) to a pin. Can be used to light a LED at varying brightnesses or drive a motor at various speeds. After a call to analogWrite(), the pin will generate a steady square wave of the specified duty cycle until the next call to analogWrite() (or a call to digitalRead() or digitalWrite() on the same pin). The frequency of the PWM signal on most pins is approximately 490 Hz. On the Uno and similar boards, pins 5 and 6 have a frequency of approximately 980 Hz.. Syntax:analogWrite(pin, value);

3. **delay() -** Pauses the program for the amount of time (in milliseconds) specified as parameter. (There are 1000 milliseconds in a second.) Syntax:delay(ms);

Pulse Width Modulation, or PWM, is a technique for getting analog results with digital means. Digital control is used to create a square wave, a signal switched between on and off. This on-off pattern can simulate voltages in between full on (5 Volts) and off (0 Volts) by changing the portion of the time the signal spends on versus the time that the signal spends off. The duration of "on time" is called the pulse width.

Code:

```
int t1=A0; //IR sensor
int t2=A2; //IR sensor
int pin5=5; //Motor pin
int pin6=6; //Motor pin
int pin7=8; //Motor pin
int pin8=7; //Motor pin
int irPin=4; //PWM Receiver pin

void setup() {
pinMode(pin5,OUTPUT);
pinMode(pin6,OUTPUT);
pinMode(pin7,OUTPUT);
pinMode(pin8,OUTPUT);
pinMode(t1,INPUT);
pinMode(t2,INPUT);
Serial.begin(9600);
}

void stopBuggy(){
        digitalWrite(pin5,LOW);
        digitalWrite(pin6,LOW);
        digitalWrite(pin7,LOW);
        digitalWrite(pin8,LOW);
}

void loop(){
int r1=digitalRead(t1);
int r2=digitalRead(t2);
if(r1==LOW && r2==LOW)
{
        digitalWrite(pin5,HIGH);
        digitalWrite(pin6,LOW);
        digitalWrite(pin7,HIGH);
        digitalWrite(pin8,LOW);

}
if(r1==LOW && r2==HIGH)
{
        digitalWrite(pin5,HIGH);
        digitalWrite(pin6,LOW);
        digitalWrite(pin7,LOW);
        digitalWrite(pin8,LOW);

}
if(r1==HIGH && r2==LOW)
{
        digitalWrite(pin5,LOW);
        digitalWrite(pin6,LOW);
        digitalWrite(pin7,HIGH);
        digitalWrite(pin8,LOW);
     }
if(r1==HIGH && r2==HIGH)
{
        digitalWrite(pin5,HIGH);
        digitalWrite(pin6,LOW);
        digitalWrite(pin7,HIGH);
        digitalWrite(pin8,LOW);

}

if (digitalRead(irPin)==HIGH)
{
        StartTime = millis();
        d = pulseIn(irPin,HIGH);
        if(d > 500 and d < 1500)
             {
                stopBuggy();
                delay(1000);
             }
}
}
```

Result Analysis:

From this experiment we get to understand the working of the gantry circuit along receiver circuit.

We send the pulse through the computer. The receiver circuit which detects the pulse via MBD311D diode and feeds it to the computer, in the form of a Square Wave Function from which the time of the high pulse is obtained using pulseIn function along with Serial.begin(9600) (which gives communication rate). And from this time along with its condition statements we operate the buggy.

Experiment 3

Objective : Write configuration program to demonsrate Xbee module communication between two PC's using XCTU.

Hardware Used: NVIS Robocar, XBee Module, USB Cable, XBee adapter

Software Used: Digi's XCTU

Theory:

Digi XBee is the brand name of a family of form factor compatible radio modules from Digi International. These are based on the IEEE 802.15.4-2003 standard designed for point-to-point and star communications at over-the-air baud rates of 250 kbit/s. XBee modules are embedded solutions providing wireless end-point connectivity to devices. These modules use the IEEE 802.15.4 networking protocol for fast point-to-multipoint or peer-to-peer networking. They are designed for high-throughput applications requiring low latency and predictable communication timing.

Code:

```
#include<NewPing.h>
#define TRIGGER_PIN 13
#define ECHO_PIN 12
#define MAX_DISTANCE 200
NewPing sonar(TRIGGER_PIN,ECHO_PIN,MAX_DISTANCE);
int count = 0;
int currentTime = 0;
int d=0;
int flag=0;
void setup() {
  pinMode(5,OUTPUT);
  pinMode(6,OUTPUT);
  pinMode(7,OUTPUT);
  pinMode(8,OUTPUT);
  pinMode(A0,INPUT);
  pinMode(A1,INPUT);
  pinMode(A2,INPUT);
  Serial.begin(9600);

}
void forward() {
  digitalWrite(5,HIGH);
  digitalWrite(6,LOW);
  digitalWrite(7,LOW);
  digitalWrite(8,HIGH);
}
void back() {
  digitalWrite(5,LOW);
  digitalWrite(6,HIGH);
  digitalWrite(7,HIGH);
  digitalWrite(8,LOW);
}
void right() {
  digitalWrite(5,LOW);
  digitalWrite(6,LOW);
  digitalWrite(7,LOW);
  digitalWrite(8,HIGH);
}
void left() {
  digitalWrite(5,HIGH);
  digitalWrite(6,LOW);
  digitalWrite(7,LOW);
  digitalWrite(8,LOW);
}
void stopp() {
  digitalWrite(5,LOW);
  digitalWrite(6,LOW);
  digitalWrite(7,LOW);
  digitalWrite(8,LOW);
}
void ultra()
{
  unsigned int distanceCm;
  distanceCm=sonar.ping_cm();
  pinMode(ECHO_PIN,LOW);
  digitalWrite(ECHO_PIN,LOW);
  pinMode(ECHO_PIN,INPUT);
  while((distanceCm<15) && (distanceCm>0))
  {
    stopp();
    distanceCm=sonar.ping_cm();
  }
}
void linefollow() {
  int l = digitalRead(A0);
  int r = digitalRead(A1);
  ultra();
  if (l && r)
  {
    forward();
  }
  if (!l && r)
  {
    left();
  }
  else if (l && !r)
  {
    right();
  }
  else if (!l && !r)
  {
    int newTime = millis();
    if (newTime - currentTime > 750)
    {
      count++;
      currentTime = newTime;
    }
    switch (count) {
      case 1: right();
      delay(400);
      Serial.println("case 1");
      break;

      case 2:
      Serial.println("case 2");
      case 3:
      Serial.println("case 3");
      forward();
      delay(150);
      break;

      case 4: forward();
      delay(50);
      right();
      delay(1400);
      Serial.println("case 4");
      break;

      case 5:
      Serial.println("case 5");
      case 6:
      Serial.println("case 6");
      forward();
      break;

      default: stopp();
      delay(1000000);

    }
  }
}
void loop()
{
  char del;
  if(flag==0)
  {
    if(Serial.available()>0)
    {
      if(del=Serial.read())
      {
        if(del=='a')
        {
          flag=1;}}}}
```

```
if(flag==1)

      linefollow();
    }
```

Experiment 4

Objective: Write a program to demonstrate full Bronze Challenge

Hardware Used: Nvis 3302ARD RoboCar Kit, XigBee modules, IR sensors, PWM transmitter and receiver

Software Used: Arduino IDE 1.8.4

Theory:

1. **Serial.begin()** - Sets the data rate in bits per second (baud) for serial data transmission. For communicating with the computer, use one of these rates: 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, or 115200.Syntax: Serial.begin(baud rate)
2. **Serial.println()** - Prints data to the serial port as human-readable ASCII text followed by a carriage return character (ASCII 13, or '\r') and a newline character (ASCII 10, or '\n'). This command takes the same forms as Serial.println(). Syntax:Serial.println(val)
3. **pinMode()** - Configures the specified pin to behave either as an input or an output. Syntax of pinMode() function is:pinMode(pin, mode);
4. **digitalWrite()** - If the pin has been configured as an OUTPUT with pinMode(), its voltage will be set to the corresponding value: 5V (or 3.3V on 3.3V boards) for HIGH, 0V (ground) for LOW. If the pin is configured as an INPUT, digitalWrite() will enable (HIGH) or disable (LOW) the internal pullup on the input pin. Syntax:digitalWrite(pin, value);
5. **digitalRead()** - Reads the value from a specified digital pin, either HIGH or LOW. Syntax:digitalRead(pin, value);
6. **delay() -** Pauses the program for the amount of time (in milliseconds) specified as parameter. (There are 1000 milliseconds in a second.) Syntax:delay(ms);

Buggy: Buggy is a robot which runs on the two wheels controlled or driven by the motor connected through the arduino board, the supply is given from the arduino board , the motors of the buggy are connected to arduino board I/O pins which can be controlled by uploading various programs of motion(forward, left, right, anticlockwise or clockwise motion) to it through Arduino IDE platform.

The program is simply sent to the arduino board through the USB.

Pulse Width Modulation, or PWM, is a technique for getting analog results with digital means. Digital control is used to create a square wave, a signal switched between on and off. This on-off pattern can simulate voltages in between full on (5 Volts) and off (0 Volts) by changing the portion of the time the signal spends on versus the time that the signal spends off. The duration of "on time" is called the pulse width.