Parking Spot Monitor

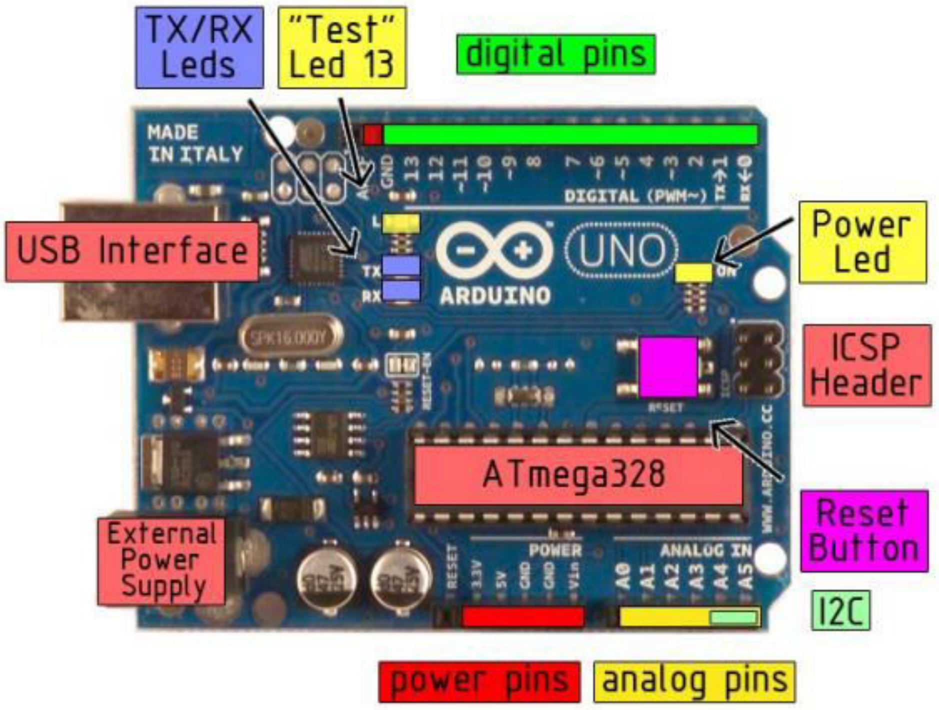
Vlad Catalin-Andrei AC CTI Anul 3

Project Description

I decided to build a parking spot monitor that uses two HC-SR04 ultrasonic sensors, one on each part of the road. One of them detects if a car passes over him and enters the parking lot, and the other detects if a car passes over him and exits the parking lot. A 7 segment display shows how many parking spots are free, based on the initial number of parking spots and the number of cars that enter or exit the parking lot. Two LEDs (green and red) indicate whether the parking lot is full or there are free parking spots. In addition, there is a barrier attached to a servo motor that lifts when a car is trying to enter the parking lot. When there are no spots left and a car is trying to enter, the red led blinks and the barrier does not lift. When all the spots are free and a car gets out of the parking lot, the green led blinks as there cannot be more free spots than the initial number.

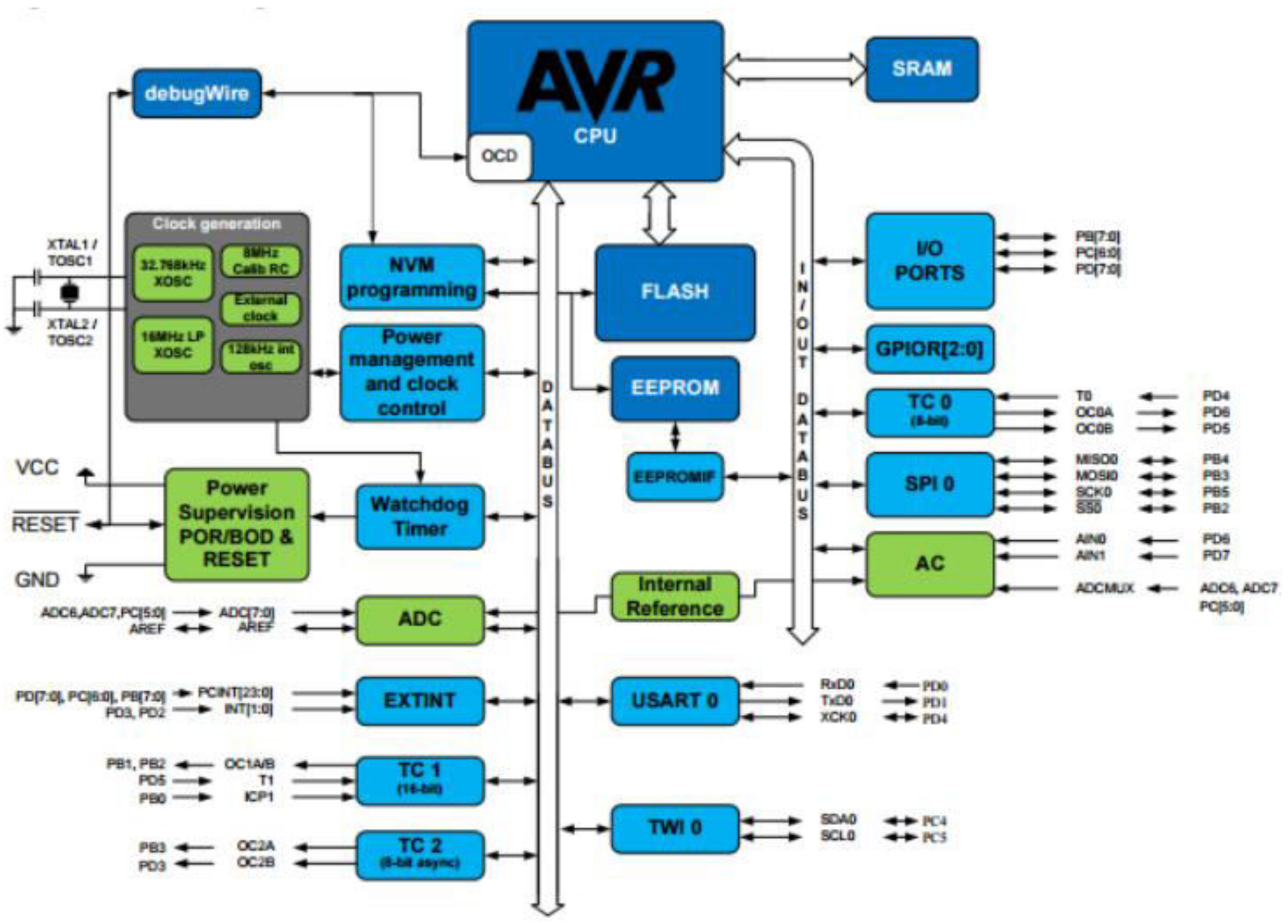
Microcontroller Description

For this project I used an Arduino Uno R3 microcontroller. It is a microcontroller board based on the ATmega328. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog input/output pins, a 16MHz clock frequency, a USB connection, a power jack, an ICSP header and a reset button. It has 32kB of Flash memory that gives enough room for most sketches



The processor of Arduino board uses Harvard architecture where the program code and program data have separate memory. It consists of two memories such as program memory and data memory. Data is stored in data memory and the code is stored in the flash program memory.

Microcontroller Modules Description



The Atmega328 microcontroller has 32 kB of flash memory for storing code (of which 0.5 KB is used for the bootloader), 2kB of SRAM, 1kB of EPROM and operates with a 16MHz clock speed. There are 32 general purpose registers, 3 timers/counters, internal and external interrupts, USUART, 2-wire serial interface, SPI port and 6-channel 10-bit A2D.

Serial: 0 (RX) and 1 (TX) are used to receive (RX) and transmit (TX) TTL serial data.

Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port C is a 7-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The PC5..0 output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running.

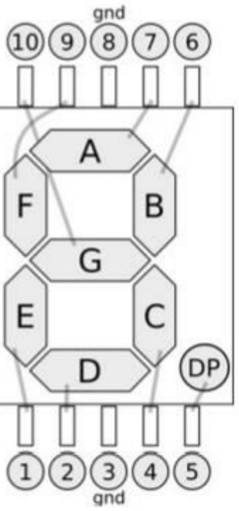
Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

AVCC is the supply voltage pin for the A/D Converter, PC3:0, and ADC7:6. It should be externally connected to VCC, even if the ADC is not used. If the ADC is used, it should be connected to VCC through a low-pass filter. Note that PC6..4 use digital supply voltage, VCC.

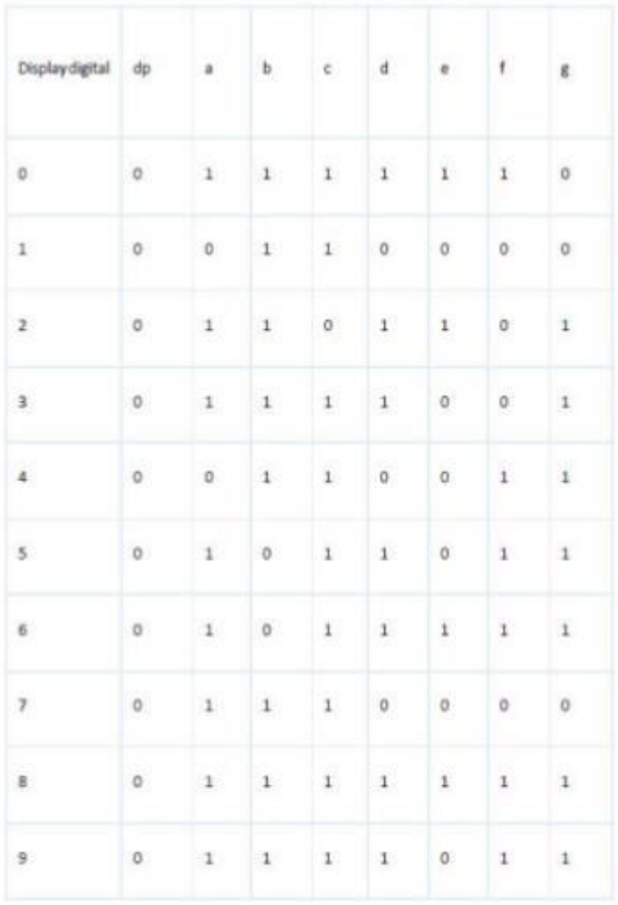
AREF is the analog reference pin for the A/D Converter.

7 Segment Display Description

7 Segment Display



0-9 ten digits correspond with each segment are as follows (the following table applies common cathode seven segment display device):



Displaydigital	dp	a	b	c	d	e	f	g
0	0	1	1	1	1	1	1	0
1	0	0	1	1	0	0	0	0
2	0	1	1	0	1	1	0	1
3	0	1	1	1	1	0	0	1
4	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1
6	0	1	0	1	1	1	1	1
7	0	1	1	1	0	0	0	0
8	0	1	1	1	1	1	1	1
9	0	1	1	1	1	0	1	1

HC-SR04 Sensor Description

Ultrasonic sensor module HC-SR04 provides 2cm to 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules includes ultrasonic transmitters, receiver and control circuit. The basic principle of work:

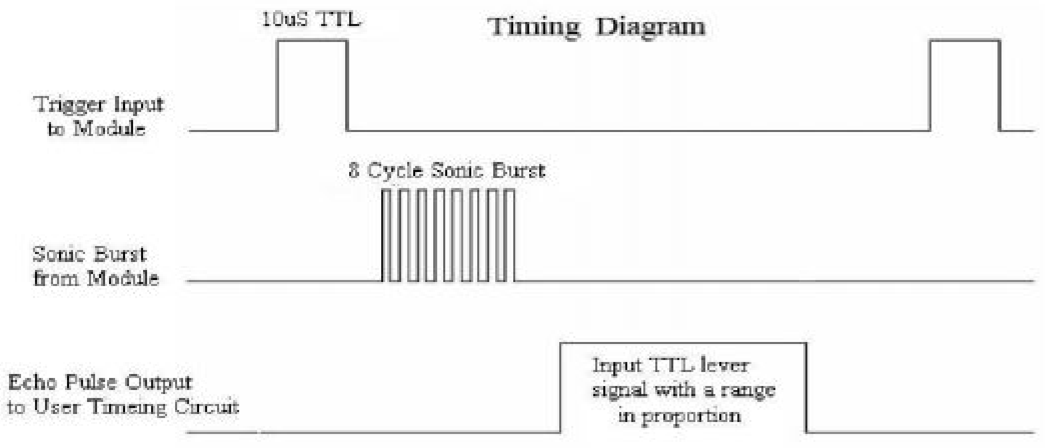
- Using IO trigger for at least 10ms high level signal,
- The module automatically sends eight 40 kHz and detect whether there is a pulse signal back.
- If the signal back, through high level , time of high output IO duration is the time from sending ultrasonic tore turning.

Test distance = (high level time × velocity of sound (340m/s) /2

Wire connecting direct as following:

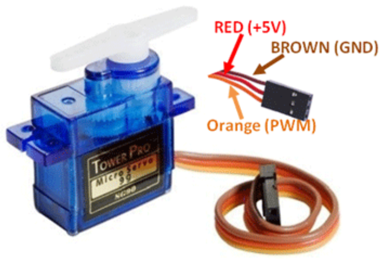
-Vcc – 5V Supply
-Trigger Pulse Output – send eight 40 kHz
-Echo Pulse Input – receive the echoed signal
-Ground

You only need to supply a short 10ms pulse to the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo. The Echo is a distance object that is pulse width and the range in proportion.


10uS TTL
Timing Diagram
Trigger Input to Module
8 Cycle Sonic Burst
Sonic Burst from Module
Echo Pulse Output to User Timeing Circuit
Input TTL lever signal with a range in proportion

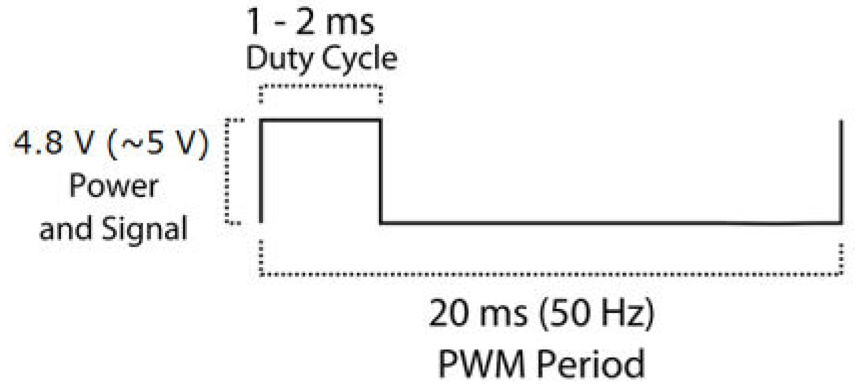
Servo Motor SG-90 Description

Servo is a type of geared motor that can only rotate 180 degrees. It is controlled by sending electrical pulses from your UNO R3 board. These pulses tell the servo what
position it should move to. The Servo has three wires, of which the brown one is the ground wire and should be connected to the GND port of UNO, the red one is the power wire and should be connected to the 5V port, and the orange one is the signal wire.



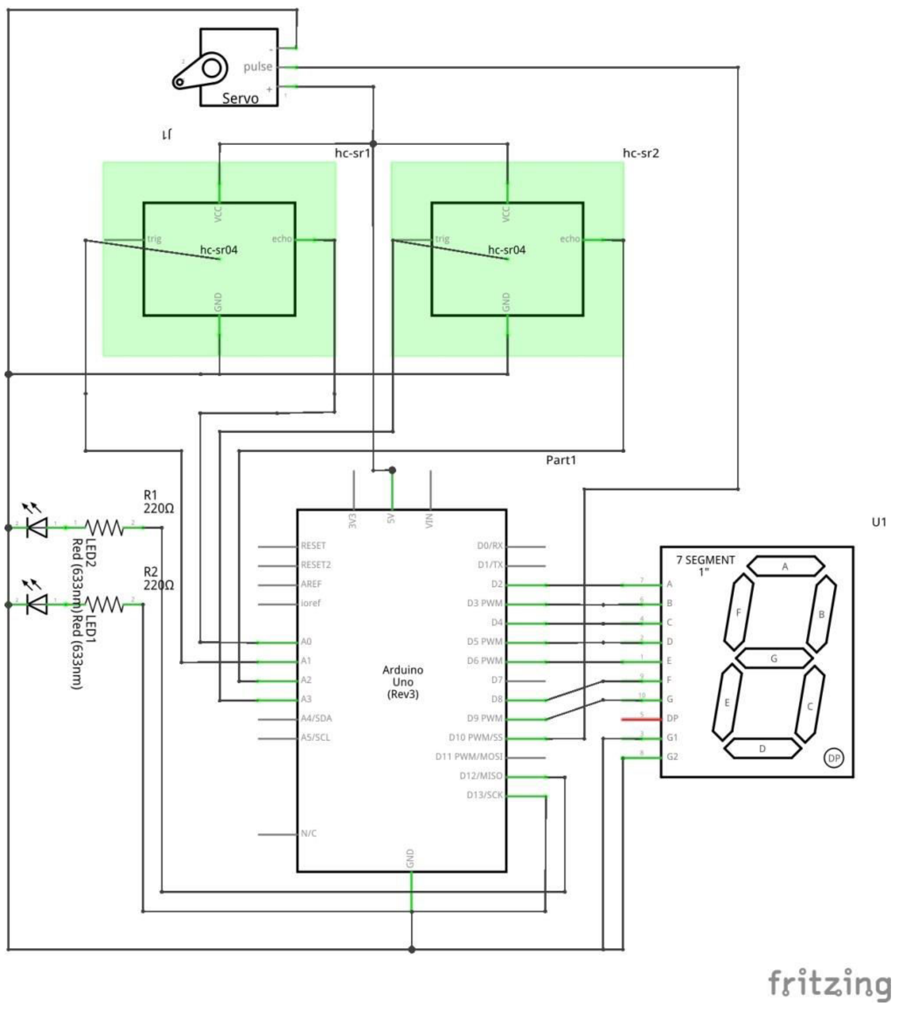
Wires:
-Brown – Ground wire connected to the ground of the system
-Red – Powers the motor, typically +5V.
-Orange – PWM signal is given in through this wire to drive the motor

The motor needs something that can generate PWM signals, this something can be anything from a 555 Timer to other microcontroller platforms like Arduino, PIC, ARM. To do this, we use an analog pin from Arduino board.



From the picture we can understand that the PWM signal produced should have a frequency of 50Hz or 20 ms period. The duty cycle can vary from 1ms to 2ms, so when it is 1ms, the motor will be in 0° and when it is 2ms it will be 180°. So, by varying the duty cycle form 1ms to 2ms the motor can be controlled from 0° to 180°.

Hardware Design



Program

```
////////////////////////////////////////////////Sketch///////////////////////////////////////////////////

#include <SoftwareSerial.h>     //Needed for console debugging
#include <Servo.h>              //Servo motor class
#include "SensorHC.h"           //My custom made class for HC-SR04 sensor

//Servo motor and its pin
Servo myBarrier;
#define servoPin 10
```

```
//LEDs for showing spot availability
#define redLed 13
#define greenLed 12

//distance sensor1 pins
#define echoPin1 A0
#define trigPin1 A1

//distance sensor2 pins
#define echoPin2 A2
#define trigPin2 A3

//7 segment display
#define a 2  //For displaying segment "a"
#define b 3  //For displaying segment "b"
#define c 4  //For displaying segment "c"
#define d 5  //For displaying segment "d"
#define e 6  //For displaying segment "e"
#define f 8  //For displaying segment "f"
#define g 9  //For displaying segment "g"

//First sensor, distance detected and previous distance detected
SensorHC sensor1(echoPin1, trigPin1);
long distance1;
long prevDistance1 = 999;

//Second sensor, distance detected and previous distance detected
SensorHC sensor2(echoPin2, trigPin2);
long distance2;
long prevDistance2 = 999;

//Integer to 7 segments display format
void segmentDisplay(int number);

//Number of parking spots
int number = 9;
```

```
void setup() {
//Attach a pin to servo motor
  myBarrier.attach(servoPin);
  myBarrier.write(150);

  //For displaying in arduino console
  Serial.begin(9600);

  //setup pins for first sensor
  pinMode(trigPin1, OUTPUT);
  pinMode(echoPin1, INPUT);

  //setup pins for second sensor
  pinMode(trigPin2, OUTPUT);
  pinMode(echoPin2, INPUT);

  //setup LED pins
  pinMode(redLed, OUTPUT);
  pinMode(greenLed, OUTPUT);

  //setup 7-segment display
  pinMode(a, OUTPUT);  //A
  pinMode(b, OUTPUT);  //B
  pinMode(c, OUTPUT);  //C
  pinMode(d, OUTPUT);  //D
  pinMode(e, OUTPUT);  //E
  pinMode(f, OUTPUT);  //F
  pinMode(g, OUTPUT);  //G
}

void loop() {
 //Ping first sensor and get distance
  sensor1.ping();
  distance1 = sensor1.getDistance();

  //Print distance received
  Serial.print("FirstSensor ");
  Serial.print(distance1);
  Serial.println("cm");
  //Check if something passed over the sensor or it's the same object
  if (distance1 <=40 && prevDistance1 >40)
  {
    if (!number)
    {
      //If it's 0 then it shouldn't decrement, so a small signal is shown
      digitalWrite(redLed, LOW);
      delay(250);
      digitalWrite(redLed, HIGH);
      delay(250);
      digitalWrite(redLed, LOW);
      delay(250);
      digitalWrite(redLed, HIGH);
    }
    else{
     //Decrease parking spot number and open the barrier
      number--;
      myBarrier.write(0);
      delay(1000);
      myBarrier.write(150);
    }
  }
  prevDistance1 = distance1;

//Choose and set the right (green or red) LED
  if (number)
  {
    digitalWrite(redLed, LOW);
    digitalWrite(greenLed, HIGH);
  }
  else
  {
    digitalWrite(redLed, HIGH);
    digitalWrite(greenLed, LOW);
  }
  segmentDisplay(number);

 //Ping second sensor and get distance
  sensor2.ping();
  distance2 = sensor2.getDistance();
  //Print distance received
  Serial.print("SecondSensor ");
  Serial.print(distance2);
  Serial.println("cm");

  //Check if something passed over the sensor or it's the same object
  if(distance2 <=40 && prevDistance2 >40)
  {
    if (number == 9)
    {
      //If it's the max number of parking spots free, then it shouldn't increment, so a
small signal is shown
      digitalWrite(greenLed, LOW);
      delay(250);
      digitalWrite(greenLed, HIGH);
      delay(250);
      digitalWrite(greenLed, LOW);
      delay(250);
      digitalWrite(greenLed, HIGH);
    }
    else
      number++;
  }
  prevDistance2 = distance2;

  //Choose and set the right (green or red) LED
  if (number)
  {
    digitalWrite(redLed, LOW);
    digitalWrite(greenLed, HIGH);
    segmentDisplay(number);
  }
  else
  {
    digitalWrite(redLed, HIGH);
    digitalWrite(greenLed, LOW);
    segmentDisplay(number);
  }
}
```

```
//integer to 7display format
void segmentDisplay(int number)
{
  switch (number)
  {
    case 0:
    {
      digitalWrite(a,HIGH);
      digitalWrite(b,HIGH);
      digitalWrite(c,HIGH);
      digitalWrite(d,HIGH);
      digitalWrite(e,HIGH);
      digitalWrite(f,HIGH);
      digitalWrite(g,LOW);
      break;
    }
    case 1:
    {
      digitalWrite(a,LOW);
      digitalWrite(b,HIGH);
      digitalWrite(c,HIGH);
      digitalWrite(d,LOW);
      digitalWrite(e,LOW);
      digitalWrite(f,LOW);
      digitalWrite(g,LOW);
      break;
    }
    case 2:
    {
      digitalWrite(a,HIGH);
      digitalWrite(b,HIGH);
      digitalWrite(c,LOW);
      digitalWrite(d,HIGH);
      digitalWrite(e,HIGH);
      digitalWrite(f,LOW);
      digitalWrite(g,HIGH);
      break;
    }
    case 3:
    {
```

```
  digitalWrite(a,HIGH);
  digitalWrite(b,HIGH);
  digitalWrite(c,HIGH);
  digitalWrite(d,HIGH);
  digitalWrite(e,LOW);
  digitalWrite(f,LOW);
  digitalWrite(g,HIGH);
  break;
}
case 4:
{
  digitalWrite(a,LOW);
  digitalWrite(b,HIGH);
  digitalWrite(c,HIGH);
  digitalWrite(d,LOW);
  digitalWrite(e,LOW);
  digitalWrite(f,HIGH);
  digitalWrite(g,HIGH);
  break;
}
case 5:
{
  digitalWrite(a,HIGH);
  digitalWrite(b,LOW);
  digitalWrite(c,HIGH);
  digitalWrite(d,HIGH);
  digitalWrite(e,LOW);
  digitalWrite(f,HIGH);
  digitalWrite(g,HIGH);
  break;
}
case 6:
{
  digitalWrite(a,HIGH);
  digitalWrite(b,LOW);
  digitalWrite(c,HIGH);
  digitalWrite(d,HIGH);
  digitalWrite(e,HIGH);
  digitalWrite(f,HIGH);
  digitalWrite(g,HIGH);
```

```
 break;
}
   case 7:
{
 digitalWrite(a,HIGH);
 digitalWrite(b,HIGH);
 digitalWrite(c,HIGH);
 digitalWrite(d,LOW);
 digitalWrite(e,LOW);
 digitalWrite(f,LOW);
 digitalWrite(g,LOW);
 break;
}
case 8:
{
 digitalWrite(a,HIGH);
 digitalWrite(b,HIGH);
 digitalWrite(c,HIGH);
 digitalWrite(d,HIGH);
 digitalWrite(e,HIGH);
 digitalWrite(f,HIGH);
 digitalWrite(g,HIGH);
 break;
}
case 9:
{
 digitalWrite(a,HIGH);
 digitalWrite(b,HIGH);
 digitalWrite(c,HIGH);
 digitalWrite(d,HIGH);
 digitalWrite(e,LOW);
 digitalWrite(f,HIGH);
 digitalWrite(g,HIGH);
 break;
}
default :
{
 digitalWrite(a,LOW);
 digitalWrite(b,LOW);
 digitalWrite(c,LOW);
```

```
      digitalWrite(d,LOW);
      digitalWrite(e,LOW);
      digitalWrite(f,LOW);
      digitalWrite(g,HIGH);
    }
  }
}

/////////////////////////////////////////////////////Sensor.h/////////////////////////////////////////////////////
#ifndef SENSORHC_H_
#define SENSORHC_H_

#if defined(ARDUINO) && ARDUINO >= 100
	#include "Arduino.h"
#else
	#include "WProgram.h"
#endif

#include <inttypes.h>

class SensorHC {
public:
	SensorHC(int echoPin, int triggerPin); //Constructor, needs two pins

	long getDistance(); //Gets the calculated distance

	void ping(); //Send the signal, receive the echo and calculate distance

private:

	int echoPin_, triggerPin_;
	long duration_, distance_;
};
#endif
```

```
/////////////////////////////////////////////////////////Sensor.cpp/////////////////////////////////////////
#include "SensorHC.h"
SensorHC::SensorHC(int echoPin, int triggerPin)
{
   echoPin_ = echoPin;
   triggerPin_ = triggerPin;
}
//Returns the distance
long SensorHC::getDistance()
{
   return distance_;
}
void SensorHC::ping()
{
   digitalWrite(triggerPin_, LOW);
   delayMicroseconds(2);

   digitalWrite(triggerPin_, HIGH);     //Transmit the signal for a short time
   delayMicroseconds(10);

   digitalWrite(triggerPin_, LOW);      //Stop the signal transmission

   duration_ = pulseIn(echoPin_, HIGH);      //Get the comeback duration
   distance_ = duration_*0.034/2;        //Calculate the distance:
duration*lightSpeed/2
}
```

Bibliography

fritzing.org
www.arduino.cc
datasheet.octopart.com
https://www.farnell.com/datasheets/1682209.pdf

Optimus Electronic Introduction to Arduino