PROGRAMMING WITH ARDUINO

Course URL:
https://sites.google.com/thapar.edu/buggy/home

Arduino Programming Basics

Comments

- Comments can be anywhere
- Comments created with // or /* and */
- Comments do not affect code
- You may not need comments, but think about the community!

Variables

Arduino Data Types	Value Assigned	Value Ranges
boolean	8 Bit	True or False
byte	8 Bit	0 to 255
char	8 Bit	-127 to 128
unsigned char	8 Bit	0 to 255
word	16 Bit	0 to 65535
unsigned int	16 Bit	0 to 65535
int	16 Bit	-32768 to 32767
long	32 Bit	-2,147,483,648 to 2,147,483,647
float	32 Bit	-3.4028235E38 to 3.4028235E38

Declaring Variables

Boolean: ***boolean variableName;***

Integer: ***int variableName;***

Character: ***char variableName;***

String: ***stringName [ ];***

Assigning Variables

Boolean: ***variableName = true;***

or ***variableName = false;***

Integer: ***variableName = 32767;***

or ***variableName = -32768;***

Character: ***variableName = ‘A’;***

or ***stringName = “SparkFun”;***

Arithmetic Operators

Assume variable A holds 10 and variable B holds 20 then –

Show Example

Operator name	Operator simple	Description	Example
assignment operator	=	Stores the value to the right of the equal sign in the variable to the left of the equal sign.	A = B
addition	+	Adds two operands	A + B will give 30
subtraction	-	Subtracts second operand from the first	A - B will give -10
multiplication	*	Multiply both operands	A * B will give 200
division	/	Divide numerator by denominator	B / A will give 2
modulo	%	Modulus Operator and remainder of after an integer division	B % A will give 0

Comparison Operators

Assume variable A holds 10 and variable B holds 20 then –

Show Example

Operator name	Operator simple	Description	Example
equal to	==	Checks if the value of two operands is equal or not, if yes then condition becomes true.	(A == B) is not true
not equal to	!=	Checks if the value of two operands is equal or not, if values are not equal then condition becomes true.	(A != B) is true
less than	<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true
greater than	>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true
less than or equal to	<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true
greater than or equal to	>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true

Boolean Operators

Assume variable A holds 10 and variable B holds 20 then –

Show Example

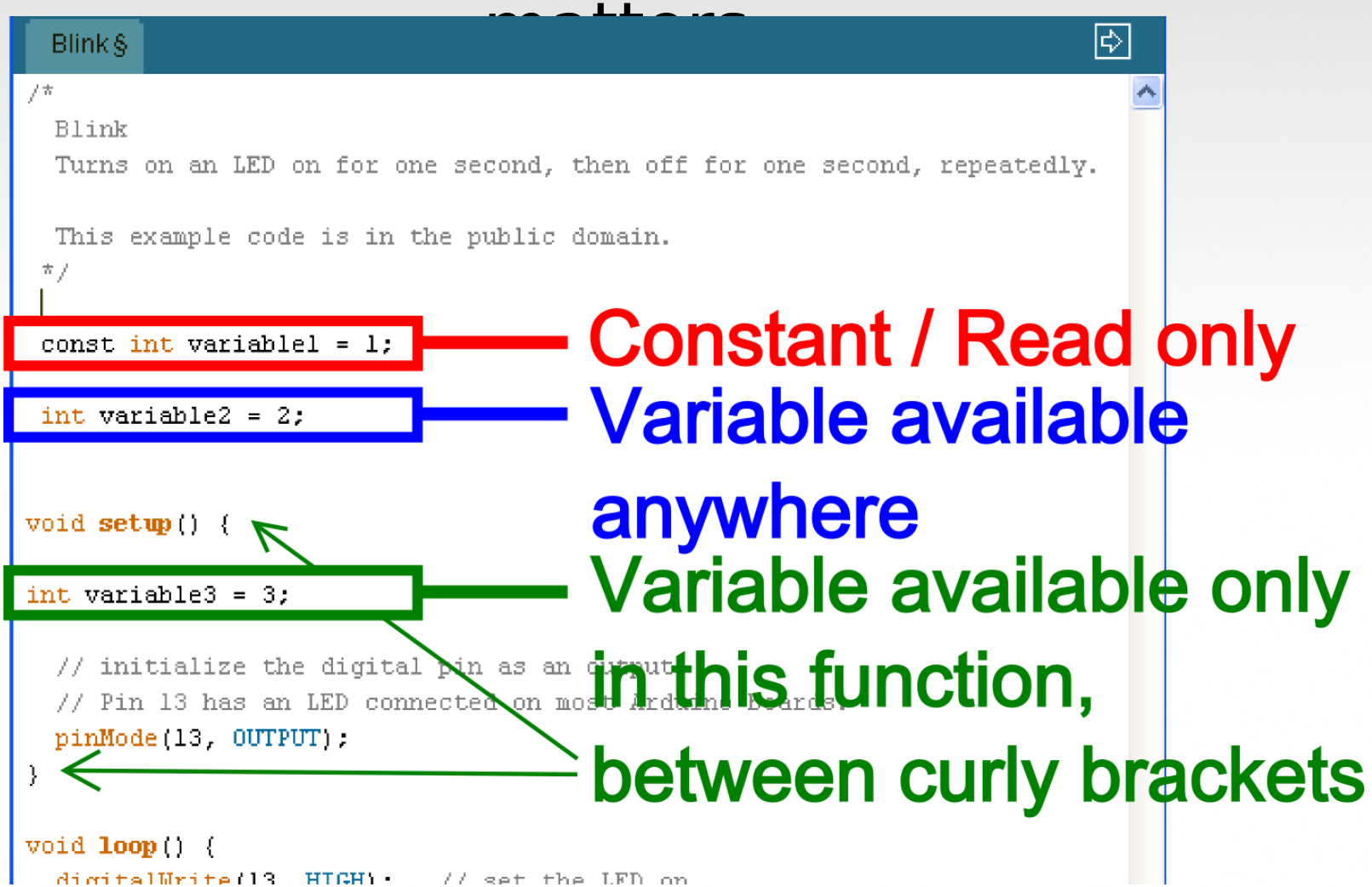
Operator name	Operator simple	Description	Example
and	&&	Called Logical AND operator. If both the operands are non-zero then then condition becomes true.	(A && B) is true
or	||	Called Logical OR Operator. If any of the two operands is non-zero then then condition becomes true.	(A || B) is true
not	!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!(A && B) is false

Compound Operators

- ++ increment
- -- decrement
- += compound addition
- -= compound subtraction
- *= compound multiplication
- /= compound division

Variable Scope

Where you declare your variables matters



Setup
void setup ( ) { }

```
void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}
```

The setup function comes before the loop function and is necessary for all Arduino sketches

Setup

void setup ( ) { }

```
void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}
```

The setup header will never change, everything else that occurs in setup happens inside the curly brackets

Setup

void setup ( ) {

pinMode (13, OUTPUT); }

```
void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}
```

Outputs are declare in setup, this is done by using the pinMode function

This particular example declares digital pin # 13 as an output, remember to use CAPS

Setup

void setup ( )

{ Serial.begin: }

```
void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
  Serial.begin(9600);
}
```

Serial communication also begins in setup

This particular example declares Serial communication at a baud rate of 9600.

Setup, Internal Pullup Resistors

void setup ( ) {

digitalWrite (12, HIGH); }

```
void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
  Serial.begin(9600);
  digitalWrite(12, HIGH);
}
```

You can also create internal pullup resistors in setup, to do so digitalWrite the pin HIGH

This takes the place of the pullup resistors currently on your circuit 7 buttons

If Statements

if (this is true) { do this; }

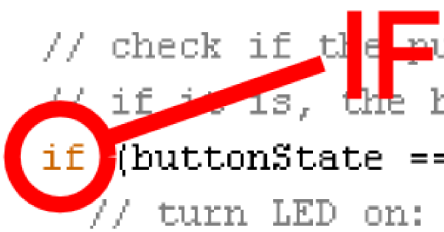
```
void loop(){
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed.
  // if it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  }
  else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

If Statement

If

if (this is true) { do this; }

```
void loop(){
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed.
  // if it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  }
  else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```



Conditional

if (this is true) { do this; }

```
void loop(){
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed.
  // if it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  }
  else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

Conditional inside parenthesis, uses ==, <=, >= or ! you can also nest using && or ||

Action

if (this is true) { do this; }

```
void loop(){
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed.
  // if it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  }
  else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

Action that occurs if conditional is true, inside of curly brackets, can be anything, even more if statements

Else

else { do this; }

```
void loop(){
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed.
  // if it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  }
  else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

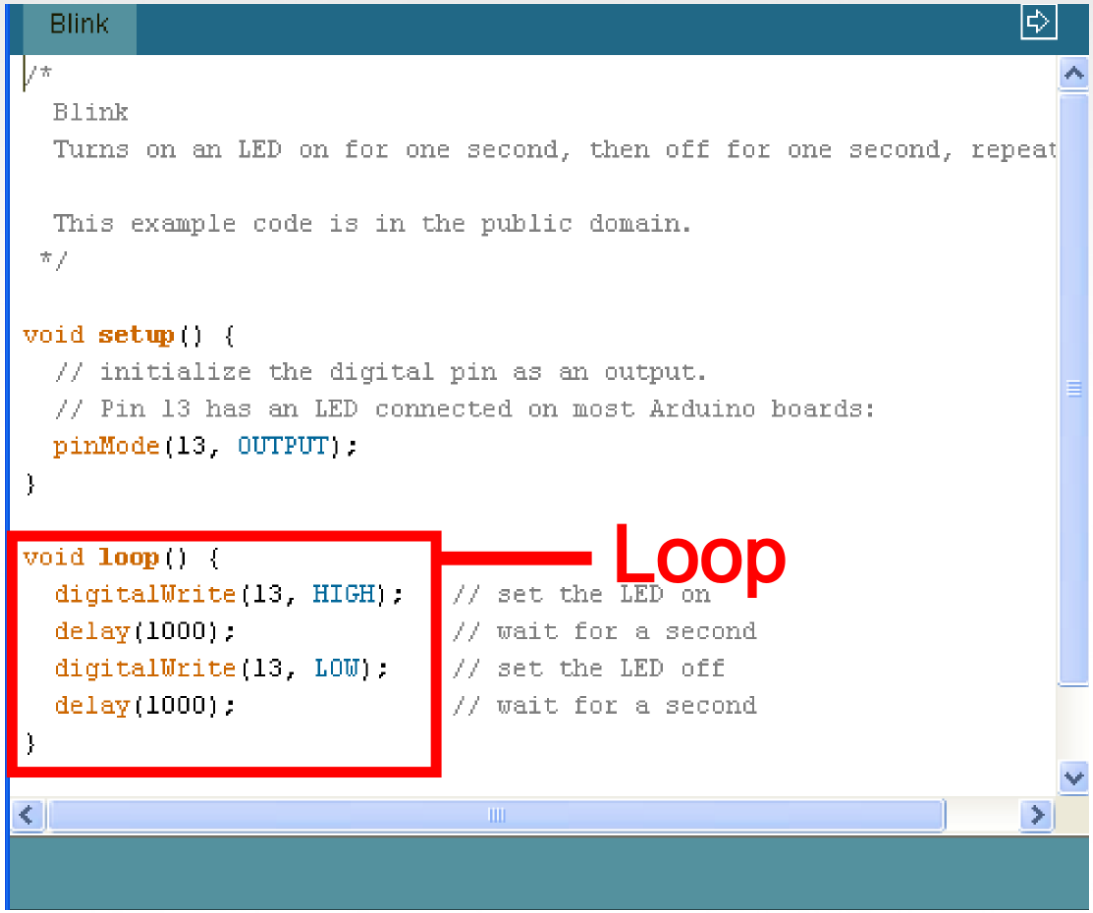
Else, optional

Basic Repetition

- loop
- For
- while

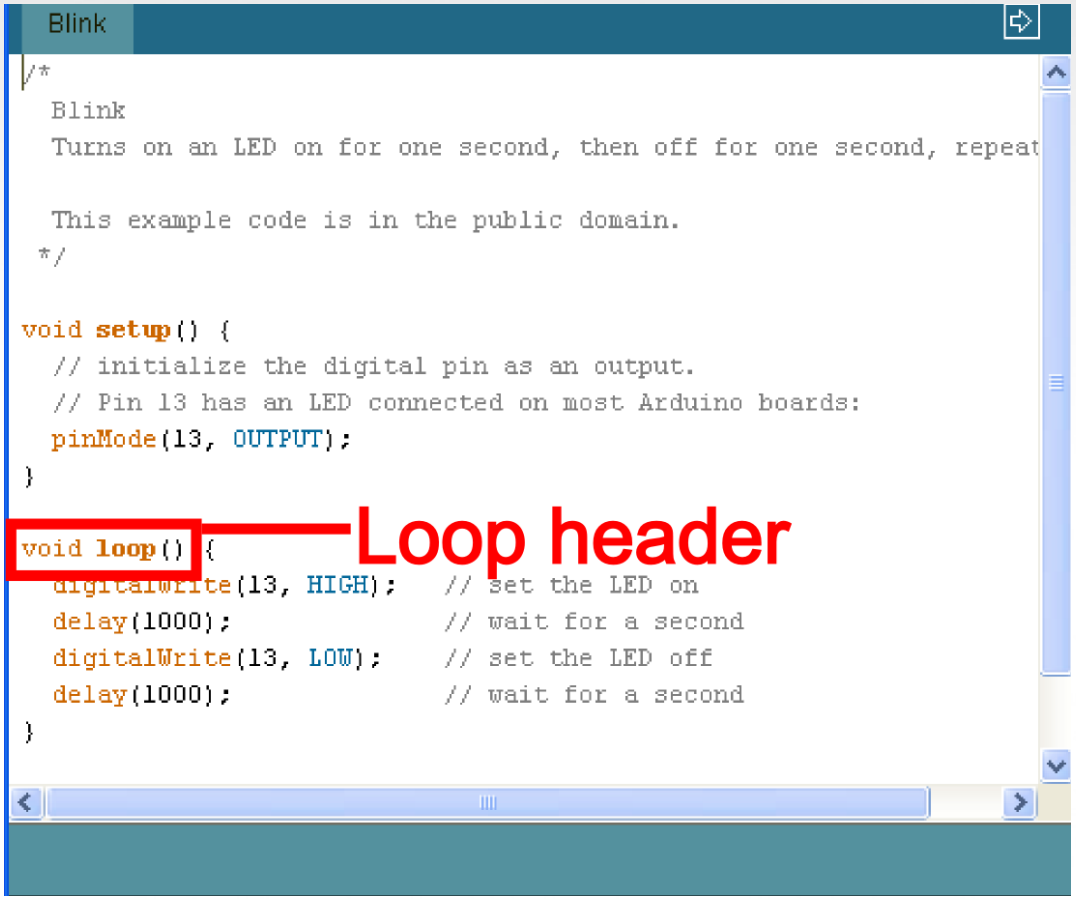
Basic Repetition

void loop ( ) { }



Basic Repetition

void loop ( ) { }



Basic Repetition

void loop ( ) { }

The “void” in the header is what the function will return (or spit out) when it happens, in this case it returns nothing so it is void

Basic Repetition

void loop ( ) { }

The “loop” in the header is what the function is called, sometimes you make the name up, sometimes (like loop) the function already has a name

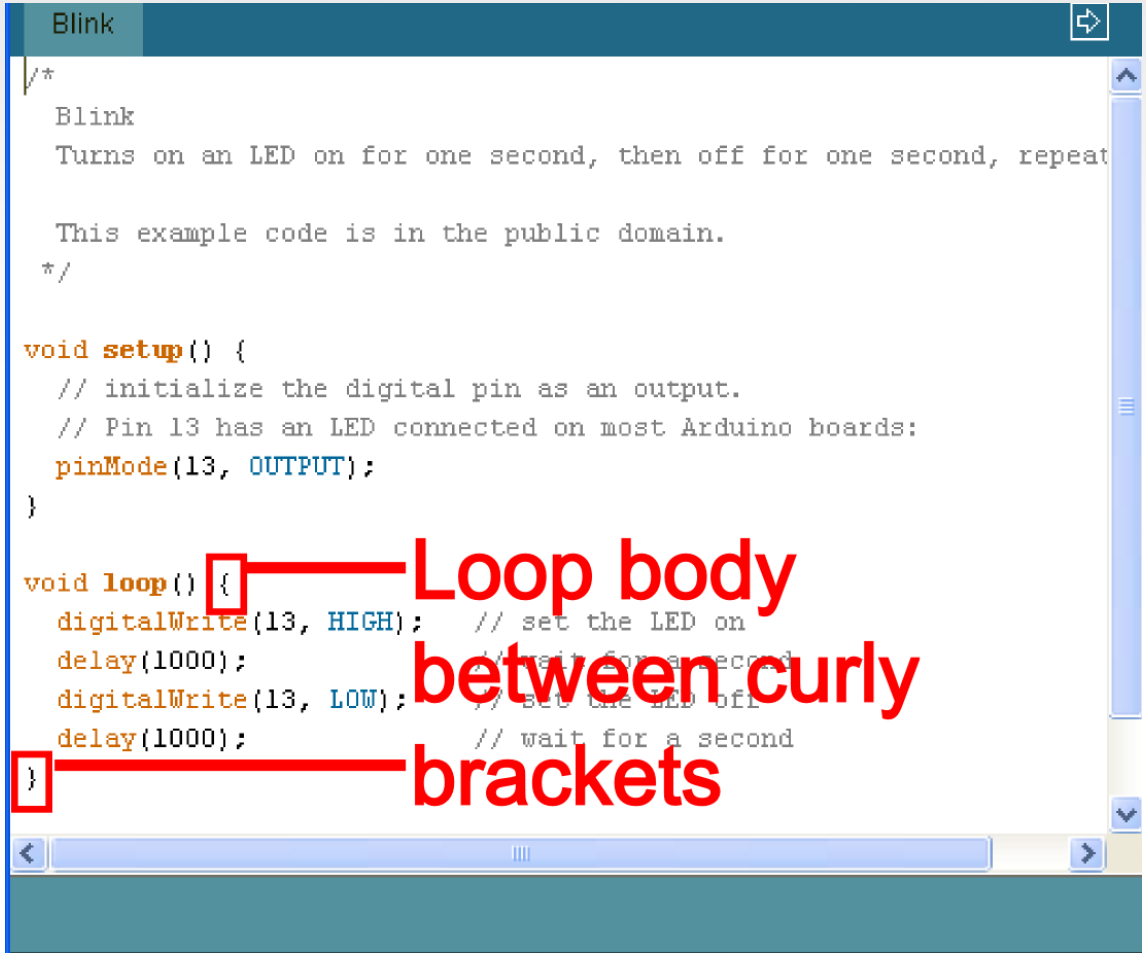
Basic Repetition

void loop ( ) { }

The “( )” in the header is where you declare any variables that you are “passing” (or sending) the function, the loop function is never “passed” any variables

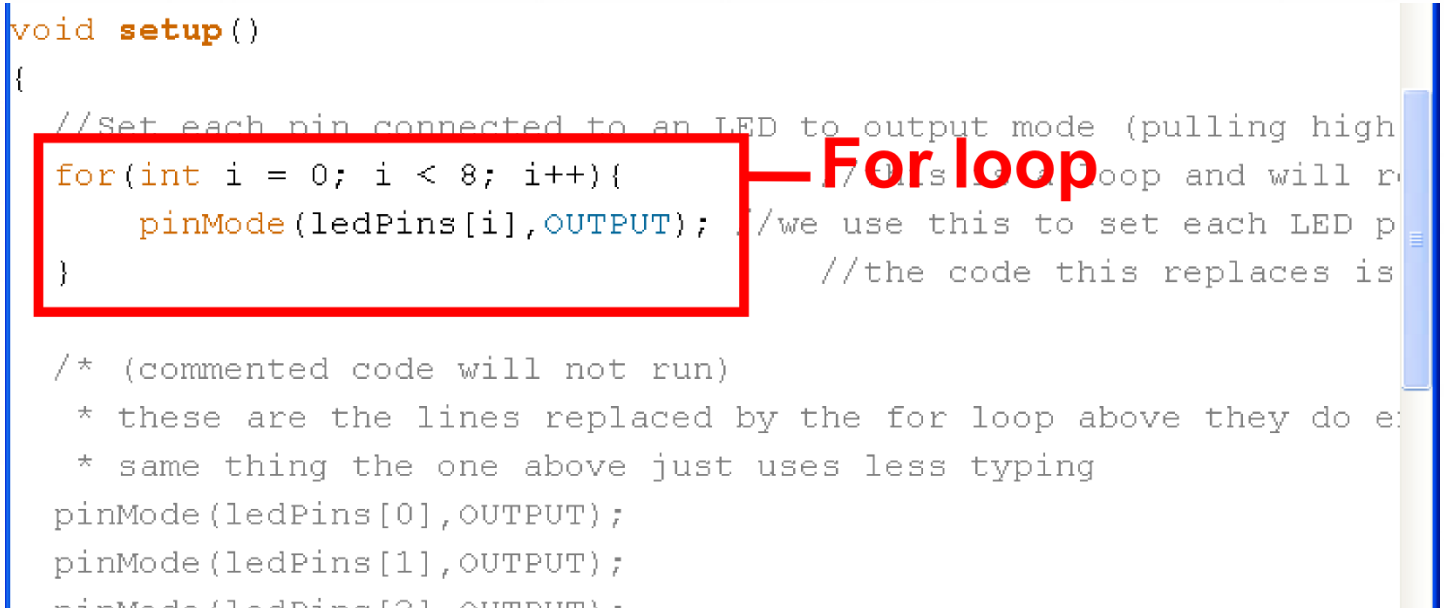
Basic Repetition

void loop ( ) { }



Basic Repetition

for (int count = 0; count<10; count++)
{
//for action code goes here
//this could be anything
}



Basic Repetition

```
while ( count<10 )
{
//while action code goes here
//should include a way to change count
//variable so the computer is not stuck
//inside the while loop forever
}
```

Basic Repetition

```
while ( count<10 )
{
//looks basically like a “for” loop
//except the variable is declared before
//and incremented inside the while
//loop
}
```

Basic Repetition
Or maybe:

```
while ( digitalRead(buttonPin)==1 )
{
//instead of changing a variable
//you just read a pin so the computer
//exits when you press a button
//or a sensor is tripped
}
```

Important functions

- Serial.println(value);
  - Prints the value to the Serial Monitor on your computer
- pinMode(pin, mode);
  - Configures a digital pin to read (input) or write (output) a digital value
- digitalRead(pin);
  - Reads a digital value (HIGH or LOW) on a pin set for input
- digitalWrite(pin, value);
  - Writes the digital value (HIGH or LOW) to a pin set for output

Using LEDs

```
void setup()
{
pinMode(13, OUTPUT);   //configure pin 13 as
output
}
// blink an LED once
void blink1()
{
digitalWrite(13,HIGH); // turn the LED on
delay(500); // wait 500 milliseconds
digitalWrite(13,LOW); // turn the LED off
delay(500); // wait 500 milliseconds
}
```

Switch Case

```
const int sensorMin = 0;      // sensor minimum, discovered through experiment
const int sensorMax = 600;    // sensor maximum, discovered through experiment

void setup() {
  // initialize serial communication:
  Serial.begin(9600);
}

void loop() {
  // read the sensor:
  int sensorReading = analogRead(A0);
  // map the sensor range to a range of four options:
  int range = map(sensorReading, sensorMin, sensorMax, 0, 3);

  // do something different depending on the range value:
  switch (range) {
    case 0:    // your hand is on the sensor
      Serial.println("dark");
      break;
    case 1:    // your hand is close to the sensor
      Serial.println("dim");
      break;
    case 2:    // your hand is a few inches from the sensor
      Serial.println("medium");
      break;
    case 3:    // your hand is nowhere near the sensor
      Serial.println("bright");
      break;
  }
  delay(1);        // delay in between reads for stability
}
```

Input & Output

- Transferring data from the computer to an Arduino is done using **Serial Transmission**
- To setup Serial communication we use the following

```
void setup() {

    Serial.begin(9600);

}
```

Writing to the Console

```
void setup() {

      Serial begin(9600);
 Serial println("Hello World!");

}

void loop() {}
```

Reading data from Arduino

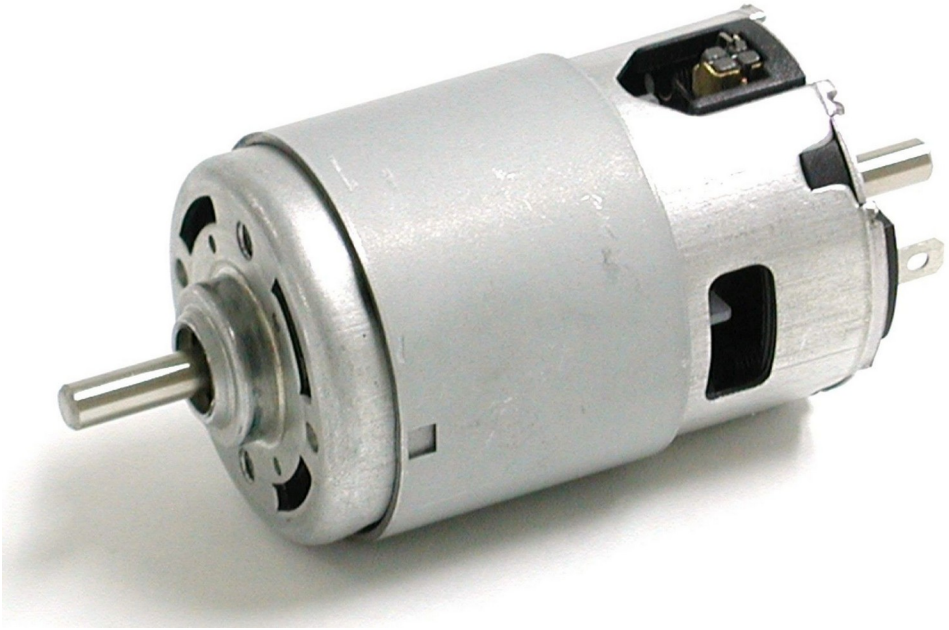
```
void setup()
{
Serial.begin(9600);
}
void serialtest()
{
int i;
for(i=0; i<10; i++)
Serial.println(i);
}
```

Arduino Code Basics

Arduino programs run on two basic sections:

```
void setup() {

        //setup motors, sensors etc

}
void loop() {

        // get information from sensors
        // send commands to motors

}
```

DC MOTORS AND ACTUATORS

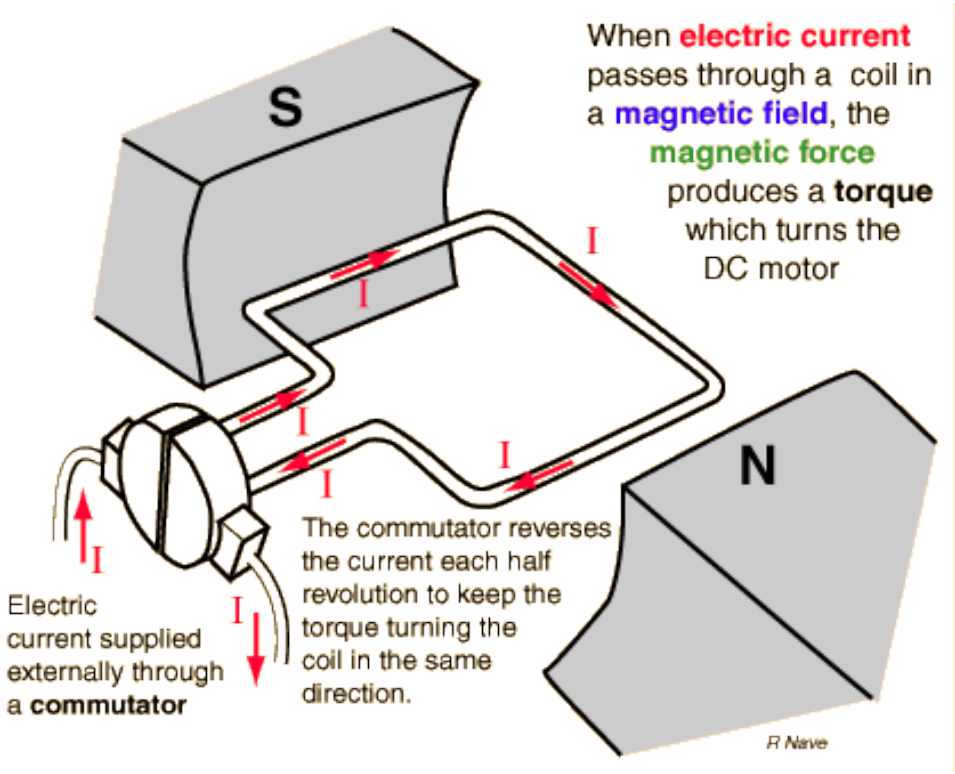


Actuators

- An electric motor is an electrical machine that converts electrical energy into mechanical energy.
- Actuator: Device that turns energy (typically electrical) to motion
- Features
  - Force
  - Speed
  - Torque
  - Efficiency

DC motor

- Force is produced due to the electric current in a wire inside a magnetic field.
- Proportional to the current, therefore can be controlled by potentiometer
- Hard to control precisely

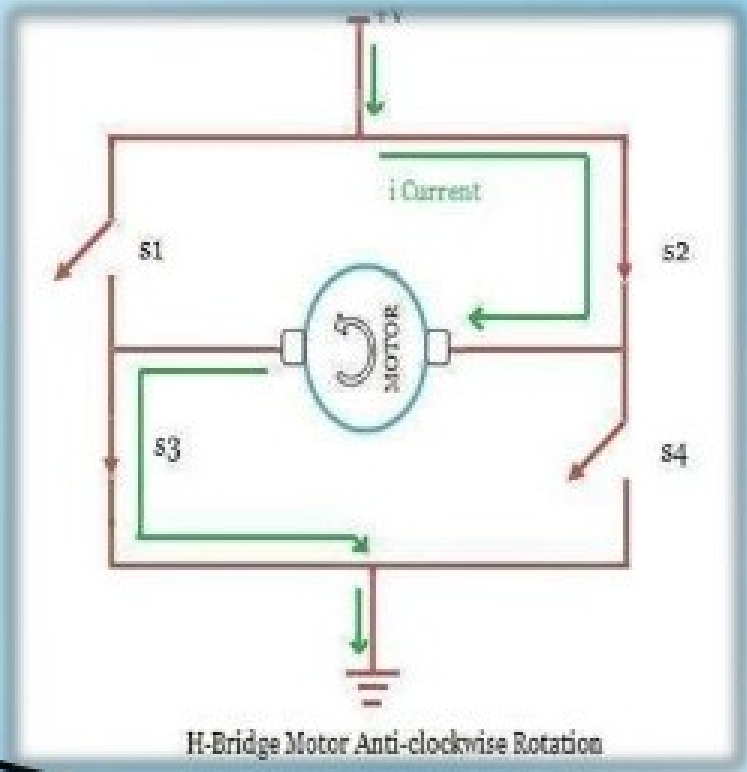


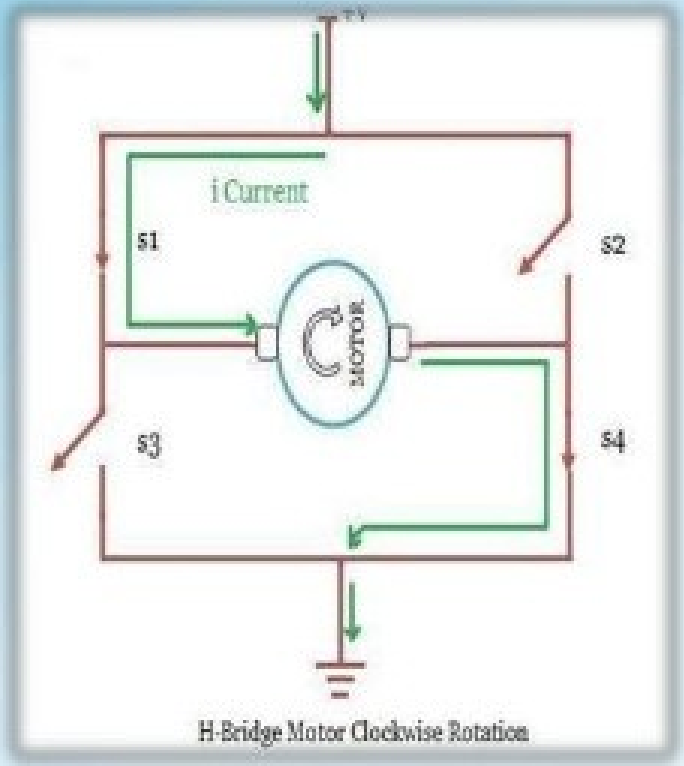
DC Motors- Motor Drivers

- Microcontrollers can supply 5V but not the enormous current required by motors.
- DC Motor Drivers act as buffer and in this case – current amplifiers.
- H – Bridge.
- Motor Driver IC-L293D.
- Different voltage can be given to motors.

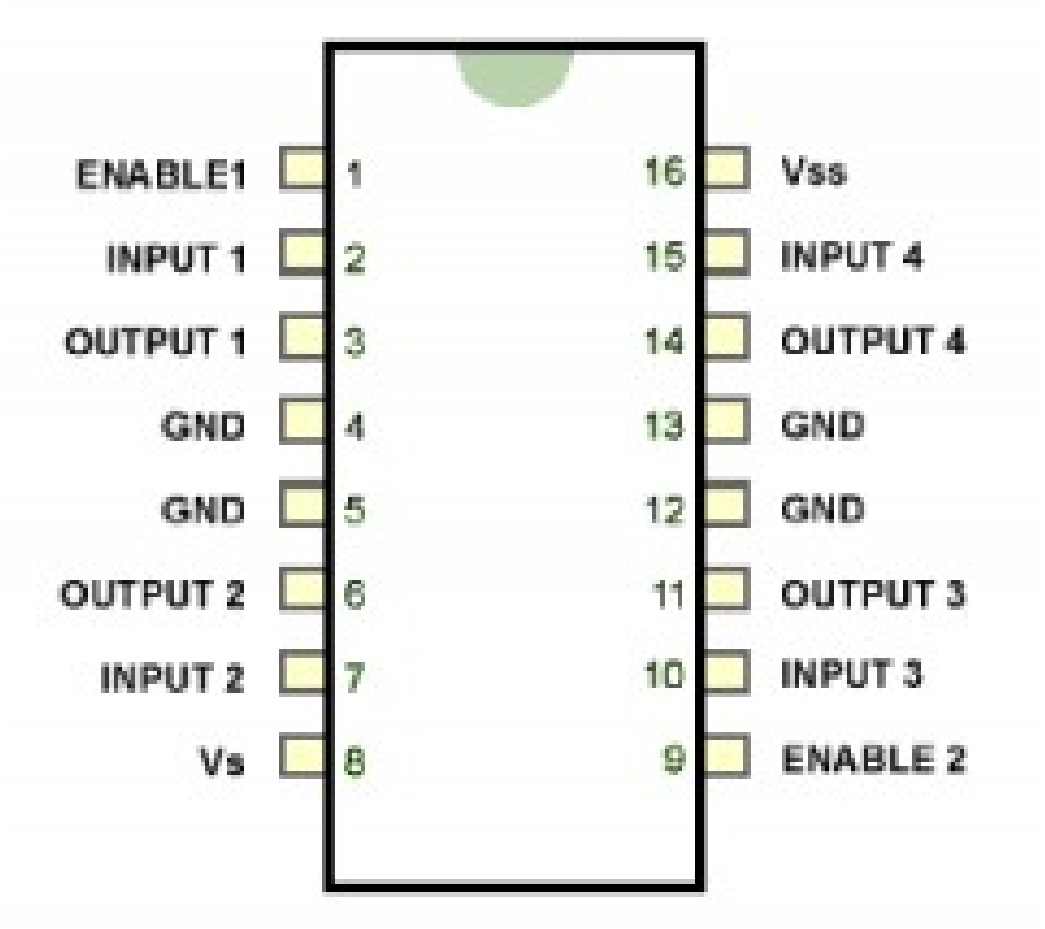
H-bridge

- Used for reversal of direction of motor
- Working

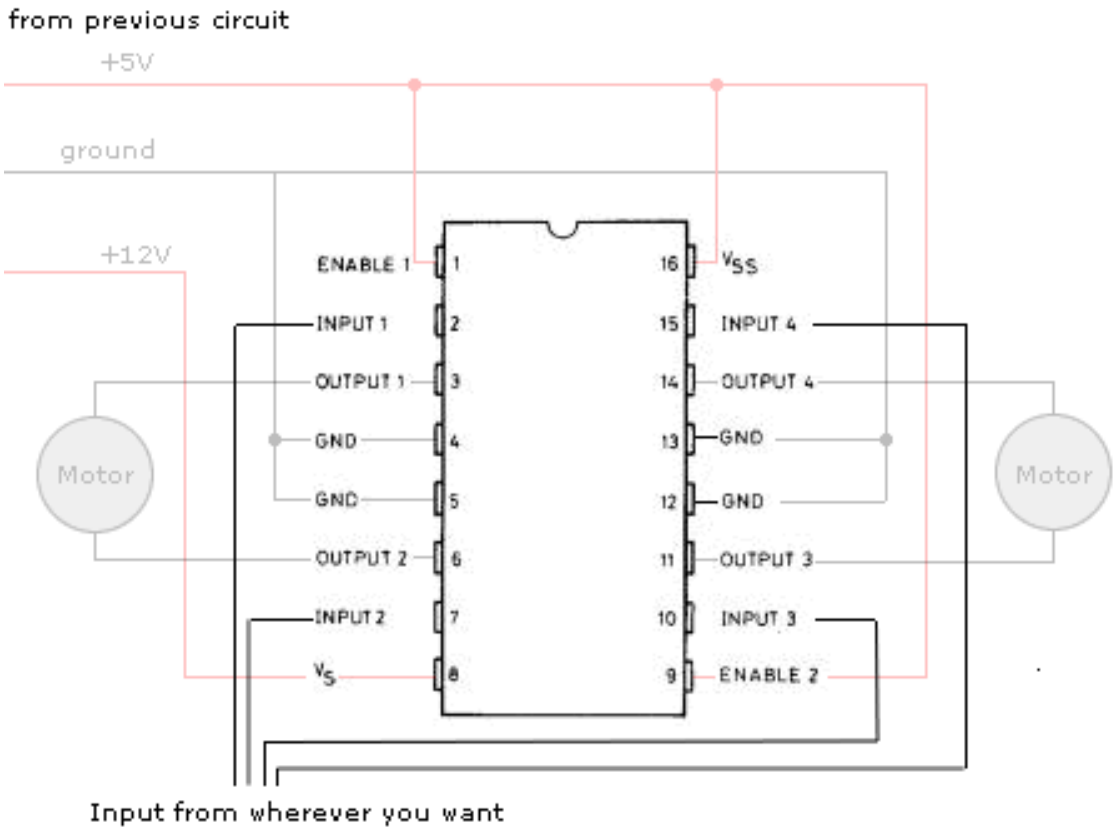




DC motor Driver/ H bridge IC/ L293D



H bridge connection with arduino



Buggy Motor Code

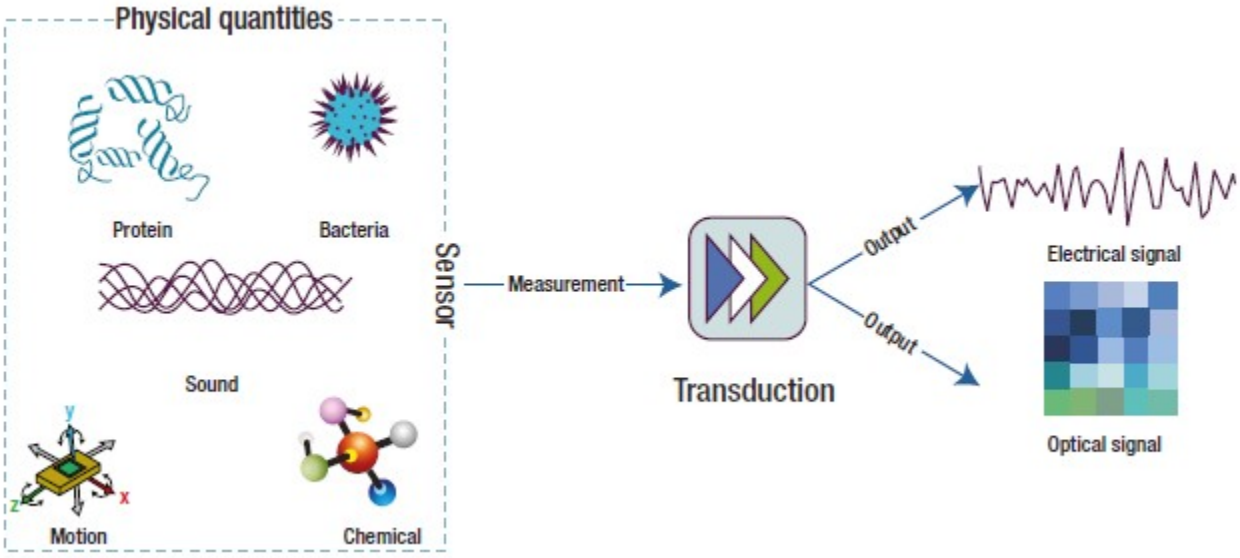
```
void setup()
    {
      pinMode(5, OUTPUT);   // Right +ve)
      pinMode(6, OUTPUT);   // Right -ve
      pinMode(7, OUTPUT);   // Left -ve
      pinMode(8, OUTPUT);   // Left +ve
    }
    void forward()
    {
      digitalWrite(5,HIGH);
    digitalWrite(6,LOW);
    digitalWrite(7,LOW);
      digitalWrite(8,HIGH);
    }
```

```
void Backward()
{
  digitalWrite(5,LOW);
  digitalWrite(6,HIGH);
  digitalWrite(7,HIGH);
  digitalWrite(8,LOW);
}
void loop()
{
  forward();
  delay(3000);    // wait for 3 second
  Backward();
  delay(3000);     // wait for 3 second
}
```

INTRODUCTION TO SENSORS

What is a sensor?

- *A device that receives a stimulus and responds with an electrical signal.*
- *A special type of transducer (device that converts one type of energy into another*



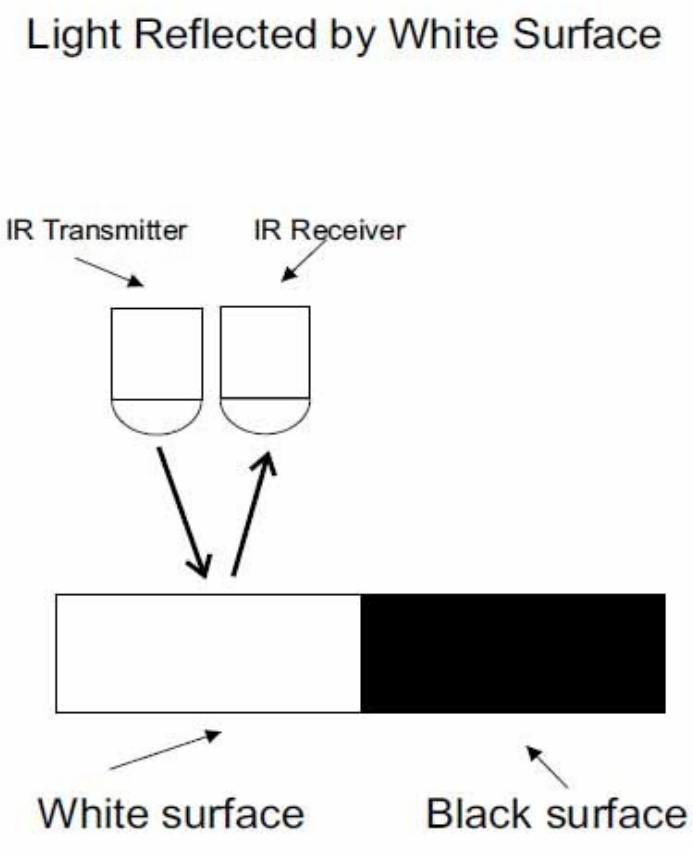
Common Sensors

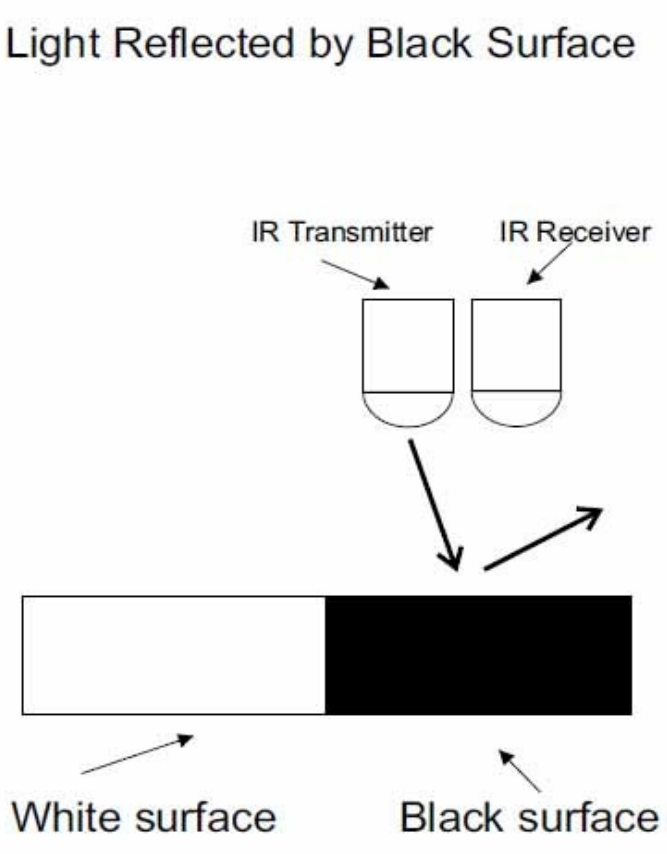
- Mechanical
  - Accelerometers: measure acceleration forces
  - Gyroscopes: useful for measuring or maintaining orientation.
- Optical
  - Photodetectors: Detect light
  - Infrared: emitting and/or detecting infrared radiation
- Semiconductor
  - Gas
  - Temperature
  - Magnetic

Sensors to be used in Buggy

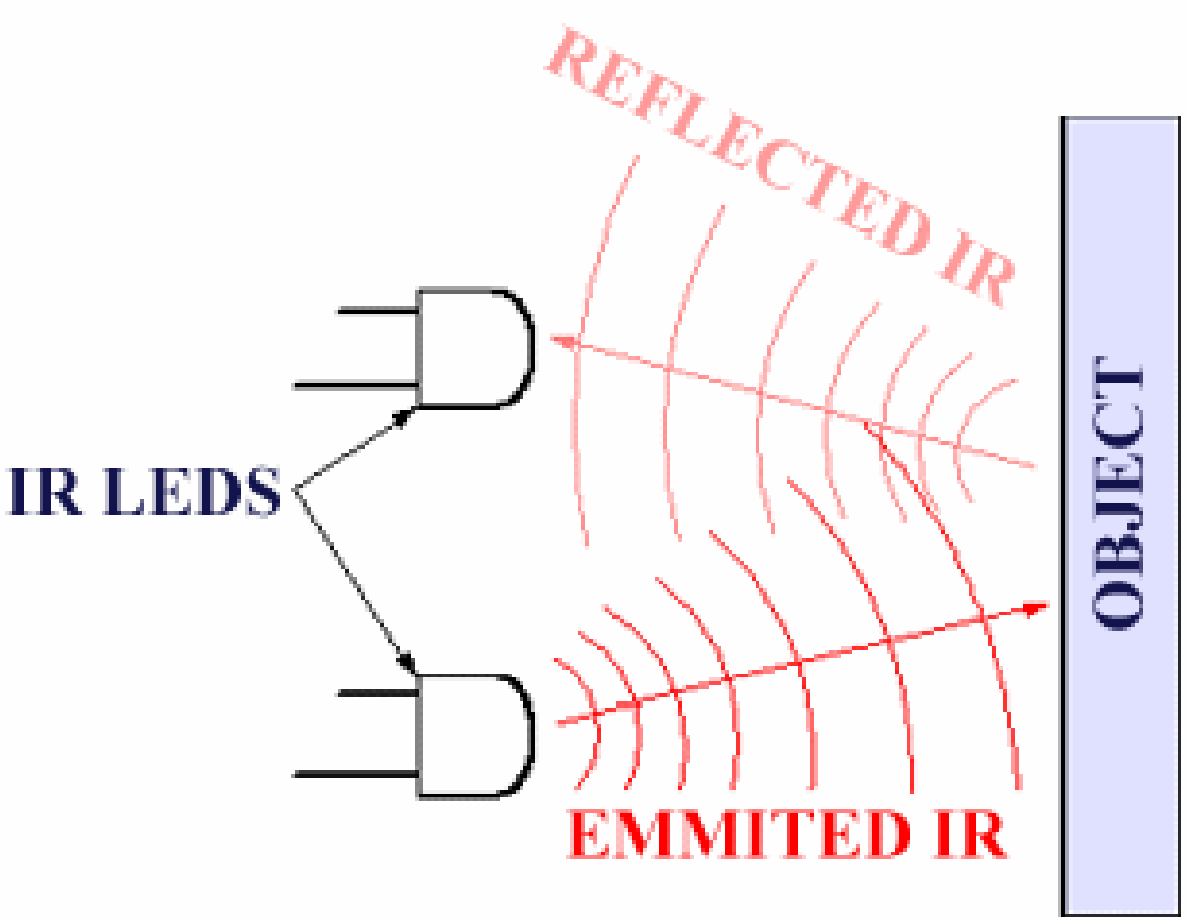
- Infra Red (IR) Sensor
- Ultrasonic Sensor
- Pulse Receiver
- ZigBee module (For inter buggy and Command centre communication)

INTRODUCTION





INTRODUCTION



WORKING

- IR-LED – Tx
- PD – Rx

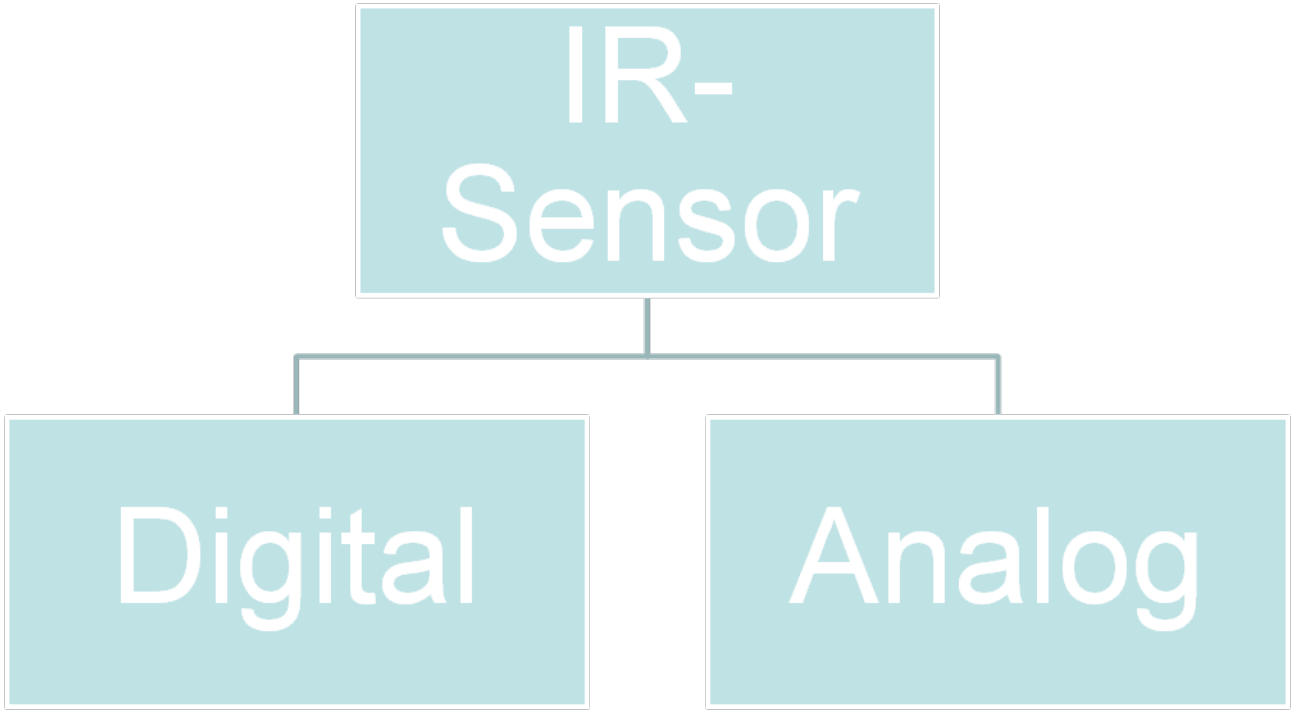
WORKING IR LED

- IR-LED work similar to normal LED
- IR-LED transmit infrared light
- IR light is not visible to our naked eye
- IR light is detected by Photodiode

WORKING PHOTODIODE

- How PD work ?
- PD generate the voltage depending on the intensity of light

TYPE OF IR-SENSOR

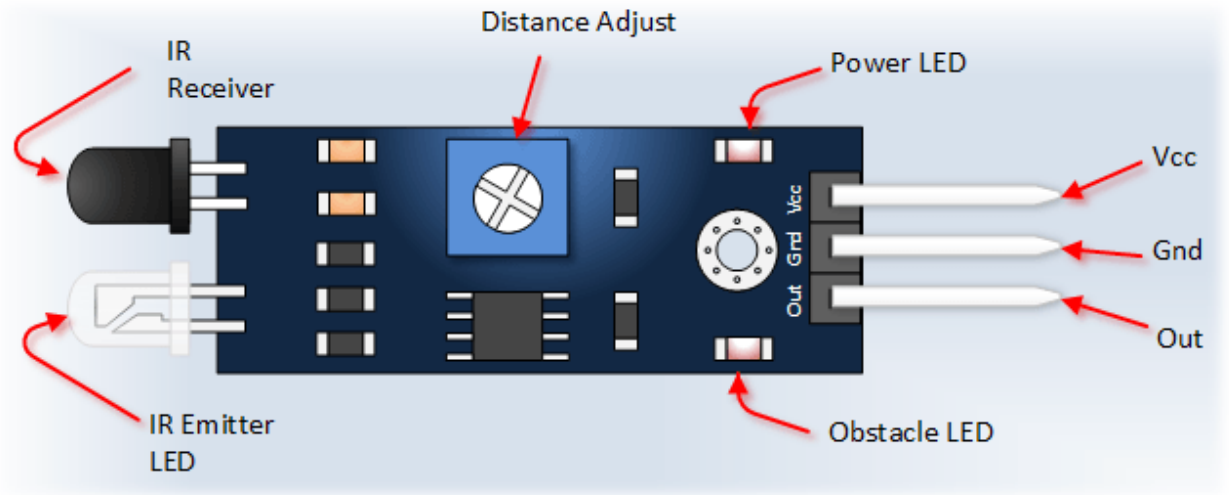


TYPE OF IR-SENSOR

- **Digital** – In digital sensor output is digital (**High** or **Low**).

- **Analog** – In analog sensor detection depend on intensity of light received.

IR-SENSOR MODULE PINOUT



In some module output is middle pin while in other its sideline pin. Extra caution is required to attach these pins to Arduino.

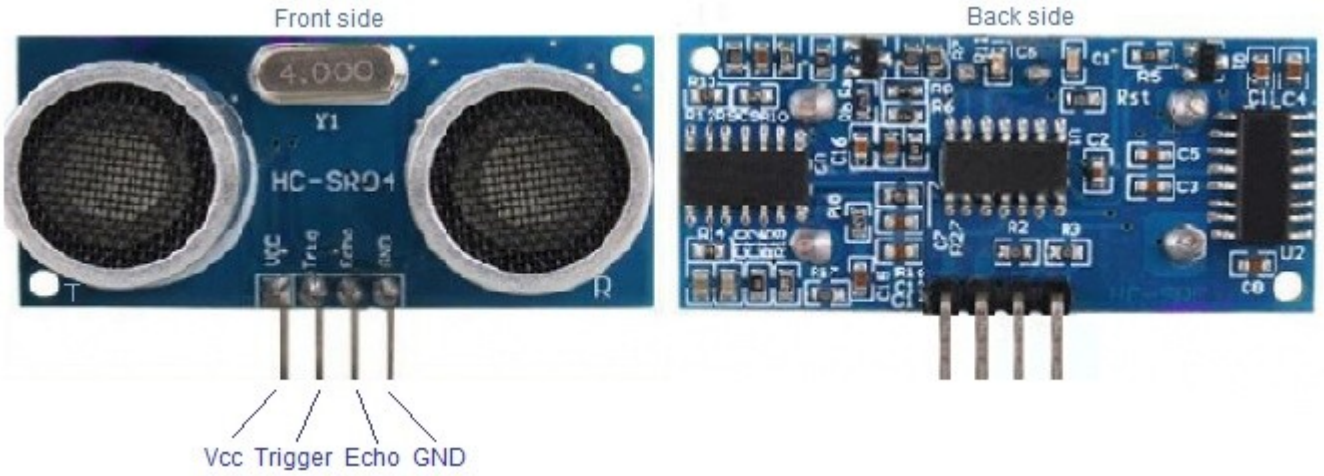
IR SENSOR IN ARDUINO

```
int r1;
int r2;
int r3;
int r4;
void setup() {
Serial.begin(9600);
pinMode(A1,INPUT);
pinMode(A3,INPUT);
}
```

```
void loop() {
r1=analogRead(A1);
r2=analogRead(A3);
r3=digitalRead(A1);
r4=digitalRead(A3);
Serial.println(r1);
Serial.print("\t");
Serial.println(r2);
Serial.print("\t");
Serial.println(r3);
Serial.print("\t");
Serial.println(r4);
}
```

ULTRASONIC SENSOR

It is essential to give your robot eyes for preventing crashes. Ultrasonic sensors, also known as transducers, have a similar working system as a radar or sonar by interpreting the echoes of radio or sound waves generated by the sensor. In this way, a robot can detect obstacles and the distance to obstacles. In this article, we make an overview of most popular ultrasonic sensors used in robotic applications and tutorials to learn how to interface and programming these sensors in order to build robots.



Hcsr04 ultrasonic sensor

Pin description:

VCC – 5V, +ve of the power supply

TRIG – Trigger Pin

ECHO – Echo Pin

GND – -ve of the power supply

HC-SR04 provides measurement function between 2 and 400 centimetres at range accuracy of 3 millimetres. The HC-SR04 module hosts the ultrasonic transmitter, the receiver and control circuit.
The HR-SR04 has four pins namely Vcc, Trigger, Echo, GND and they are explained in detail below.

1) **VCC**: 5V DC supply voltage is connected to this pin.

2) **Trigger**: The trigger signal for starting the transmission is given to this pin. The trigger signal must be a pulse with 10uS high time. When the module receives a valid trigger signal it issues 8 pulses of 40 KHz ultrasonic sound from the transmitter. The echo of this sound is picked by the receiver.

3) **Echo**: At this pin, the module outputs a waveform with high time proportional to the distance.

4) **GND**: Ground is connected to this pin.

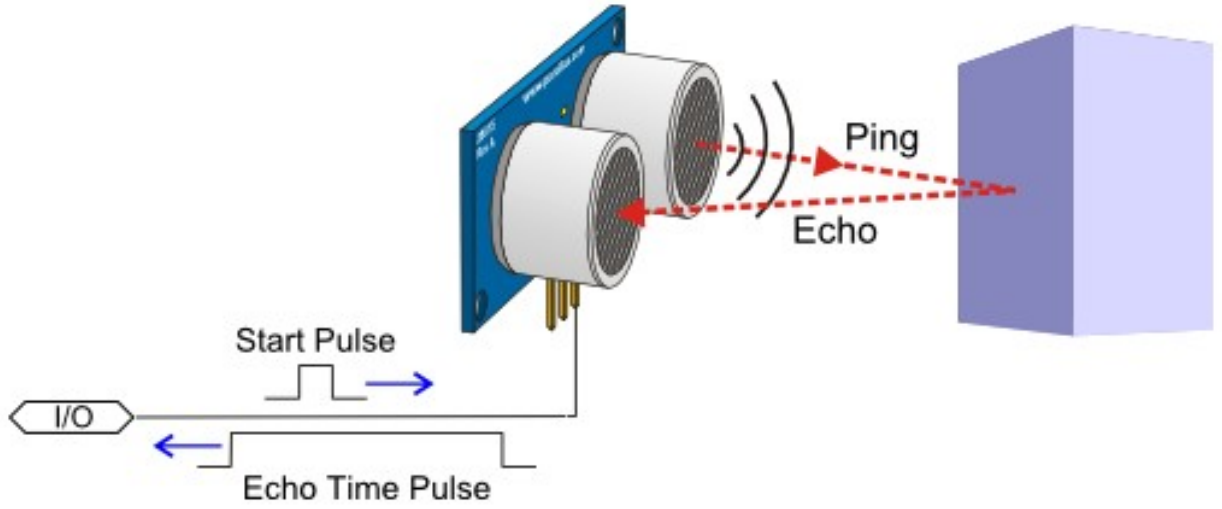
Working principle

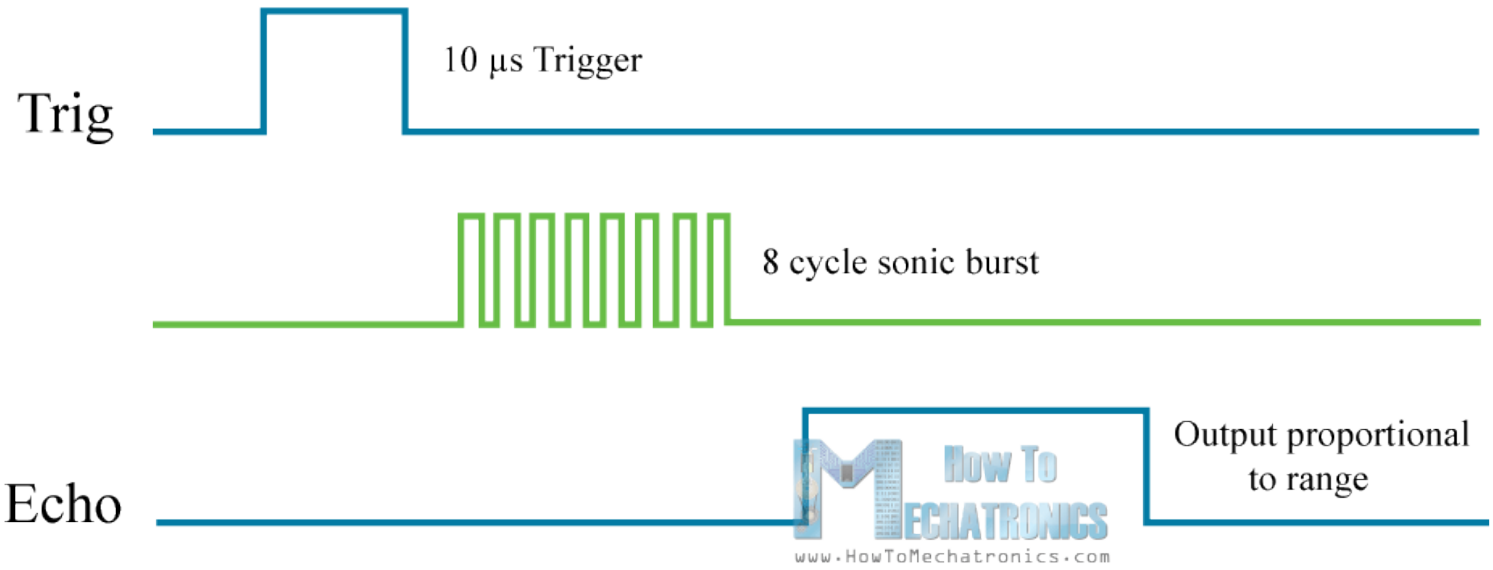
- firstly, transmit at least 10us high level pulse to the Trig pin (module automatically sends eight 40K square wave) .

- then wait to capture the rising edge output by echo port, at the same time, open the timer to start timing.

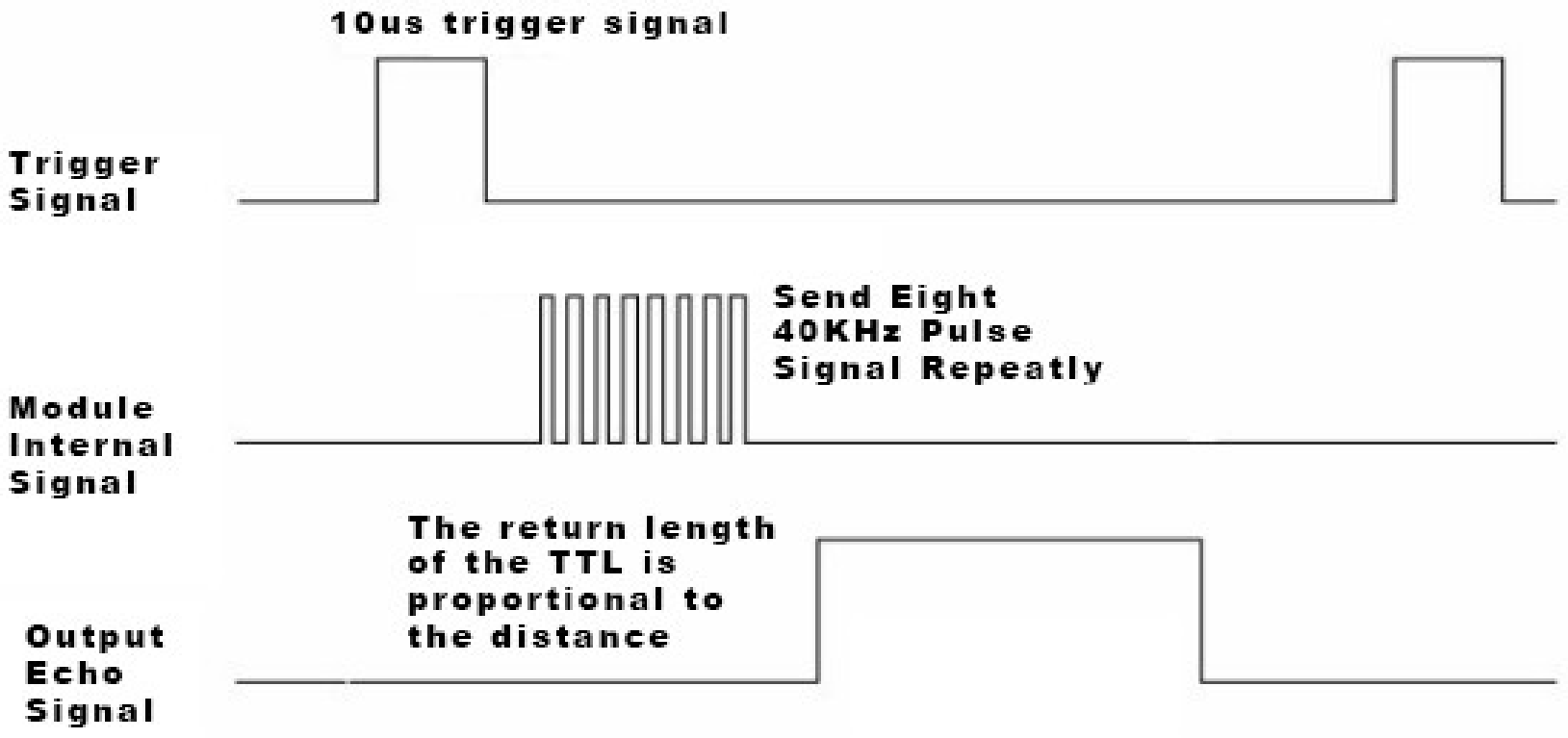
- Next, once again capture the falling edge output by echo port, at the same time, read the time of the counter, which is the ultrasonic running time in the air.

Working principle

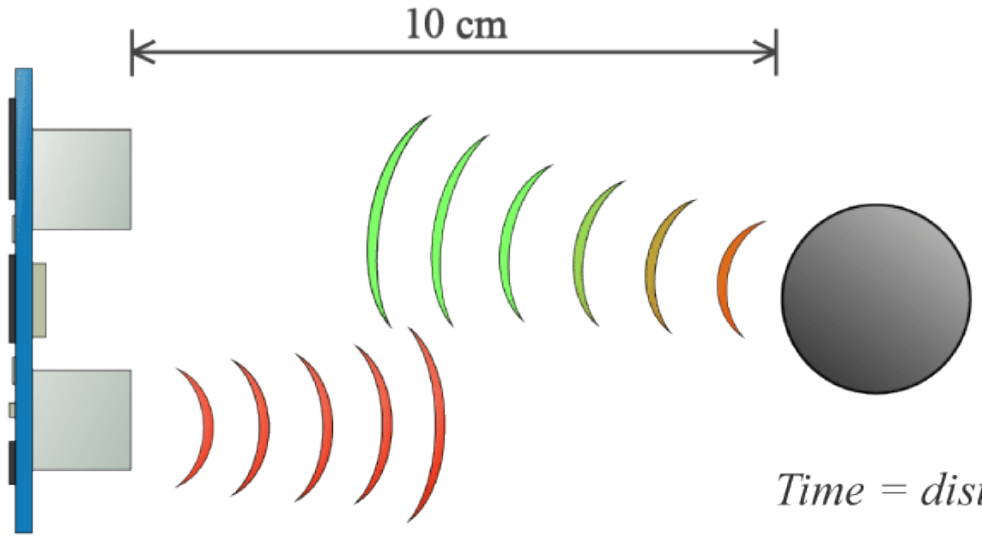




Working principle



Calculate Distance to Obstacle



speed of sound:

$$v = 340\ m/s$$

$$v = 0{,}034\ cm/\mu s$$

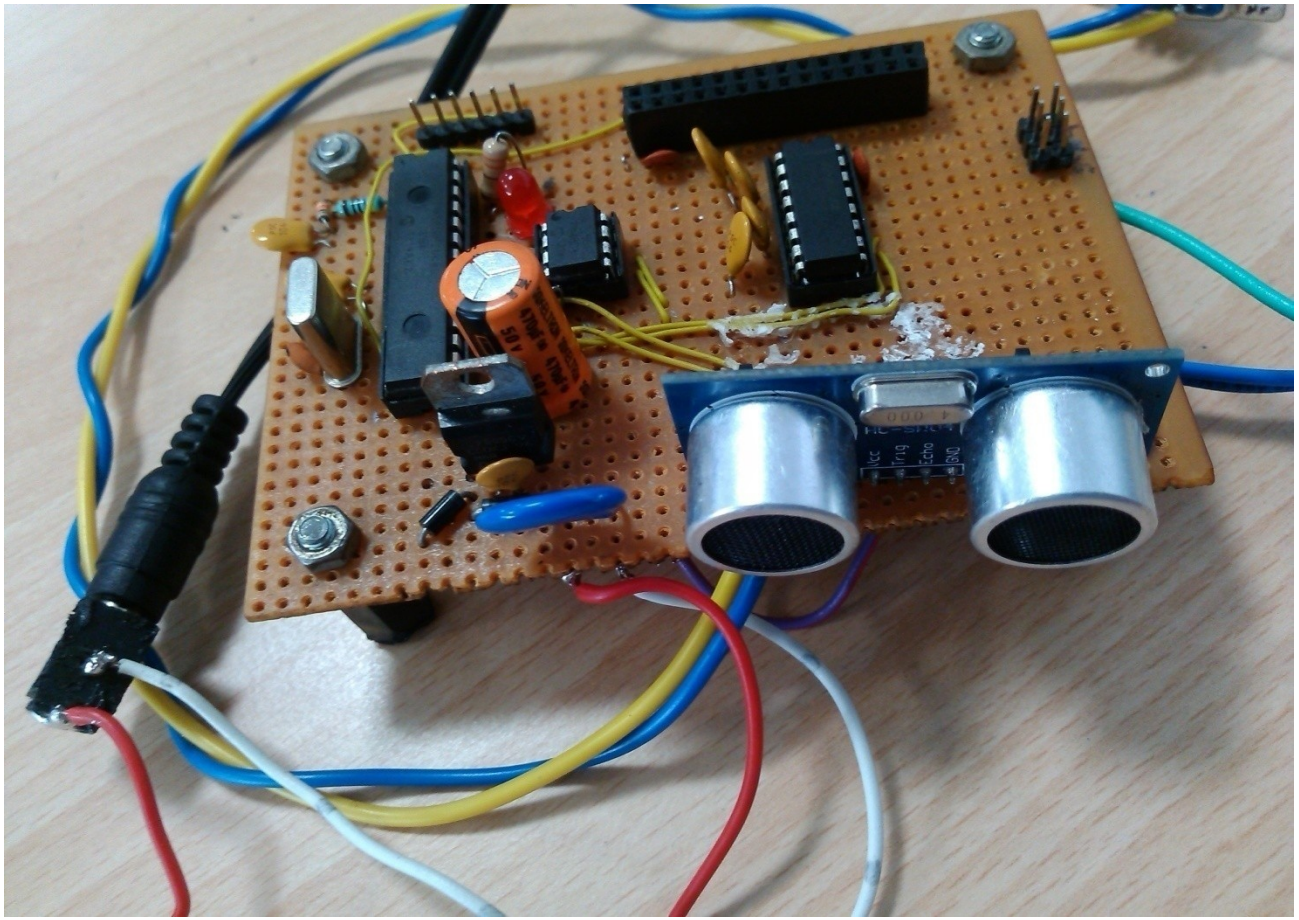
Time = distance / speed:

$$t = s / v = 10 / 0{,}034 = 294\ \mu s$$

Distance:

$$s = t \cdot 0{,}034 / 2$$

PIC18F26K80 DEVELOPMENT BOARD



Ultrasonic Sensor Example (1/2)

```
const int trigPin = 13;
const int echoPin = 12;
long duration;
int distanceCm, distanceInch;
void setup() {
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
Serial.begin(9600);
//  initialize digital pin motor as an output
  pinMode(5, OUTPUT);   // Right +ve)
  pinMode(6, OUTPUT);   // Right -ve
  pinMode(7, OUTPUT);   // Left -ve
  pinMode(8, OUTPUT);   // Left +ve
}
```

Ultrasonic Sensor Example (2/2)

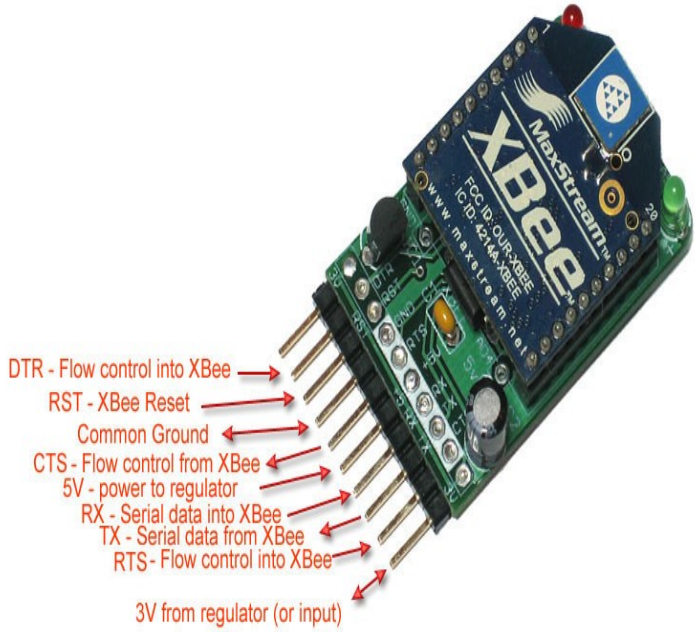
```
void loop() {
int s=0;
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);
distanceCm= (duration*0.034)/2;
distanceInch = (duration*0.0133)/2;
Serial.println(distanceCm);
if(distanceCm<30)     //here we set the object range 15cm.
{
Put your code here
}
```

Code using NewPing library

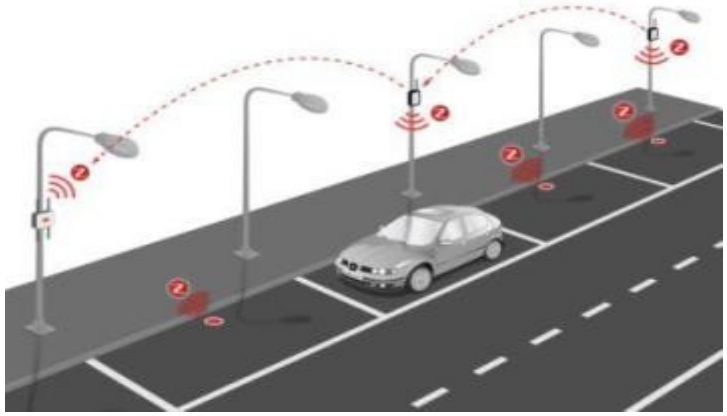
```
#include <NewPing.h>

#define TRIGGER_PIN  13
#define ECHO_PIN     12
#define MAX_DISTANCE 200
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);
void setup() {
  Serial.begin(9600);
}
void loop() {
  delay(50);
  Serial.print("Ping: ");
  Serial.print(sonar.ping_cm());
  Serial.println("cm");
}
```

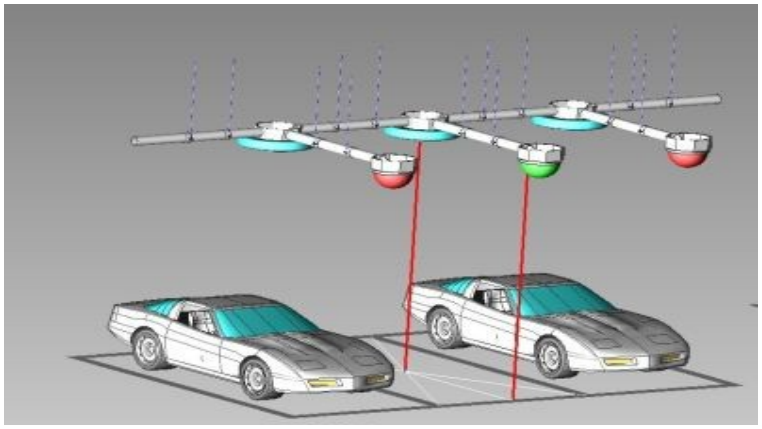
ZigBee Technology

The explosion in wireless technology has seen the emergence of many standards, especially in the industrial, scientific and medical (ISM) radio band. There have been a multitude of proprietary protocols for control applications, which bottlenecked interfacing. Need for a widely accepted standard for communication between sensors in low data rate wireless networks was felt.




Home
Control
Input
Health
ZigBee®
Energy
Telecom
Retail
Building
Light





Comparision Between Zigbee and Bluetooth

Parameter	Zigbee	Bluetooth
Power Requirements	10mA	100mA
Development Costs *(Codesize in Zigbee/Codesize in Bluetooth)*	0.5	1
Sensitivity	-92dbm	-82dbm
Number of supported nodes	65536 (mesh)	7 (star)
Security	AES (128 bit)	SAFER (64/128 bit)
Latency requirements	Optional Guaranteed Time Slot	None
Range	~75m (LOS)	10m
Maximum Data Rate	20-250 KB/s	720 KB/s
System Resources	4KB-32KB	250+KB
Battery Life (days)	100-1000+	1-7
Network Size	65K+	7
Range(m)	1-100+	1-10+