Unit-3

Sensors, Microcontrollers and their **Interfacing**

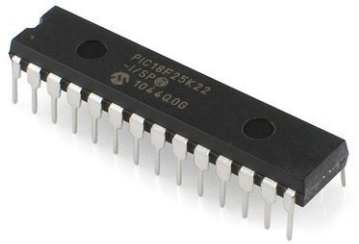
Microcontrollers

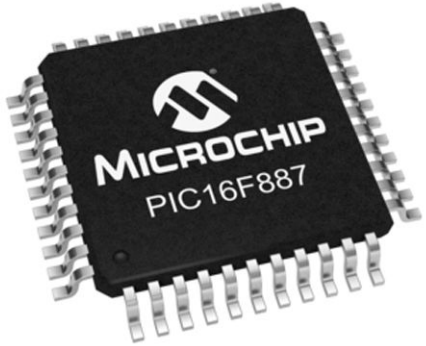
Section - 1

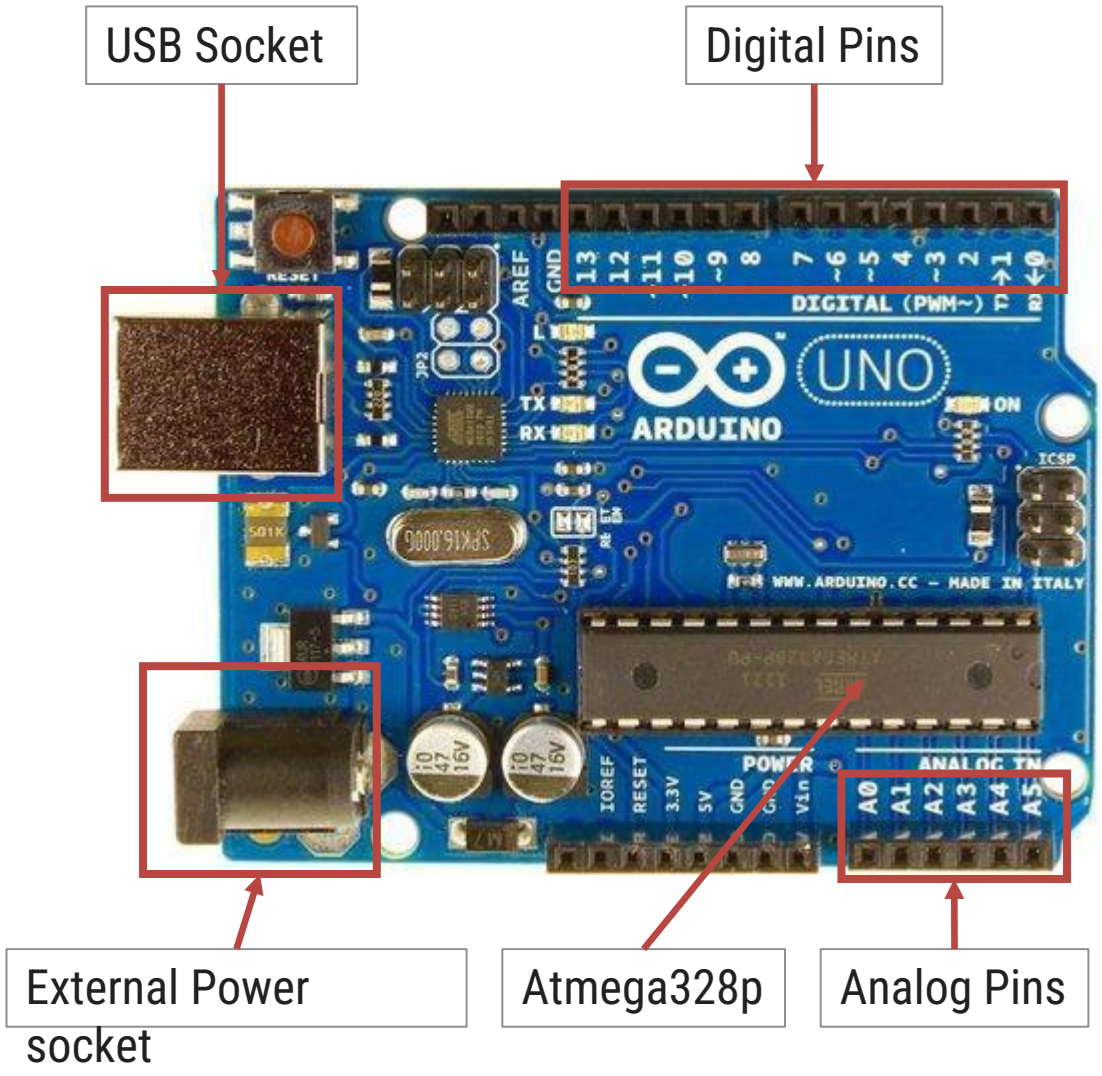
Overview of Microcontrollers

- **Microcontroller** is a digital device consisting of **CPU, peripheral I/Os, memories, interrupts,** etc. in a single chip or IC.
- Microcontroller is a **compact integrated circuit** designed for **a specific operation** in an embedded system.
- The examples of various microcontrollers are given as:

Atmel AVR	Microchip – PIC	Intel 8051
AtMega8	PIC18F452	AT89C51
AtMega328P	PIC16F877	AT89S52
AtMega16	PIC16F676	AT89c2051
AtMega2560	PIC16F72	P89V51

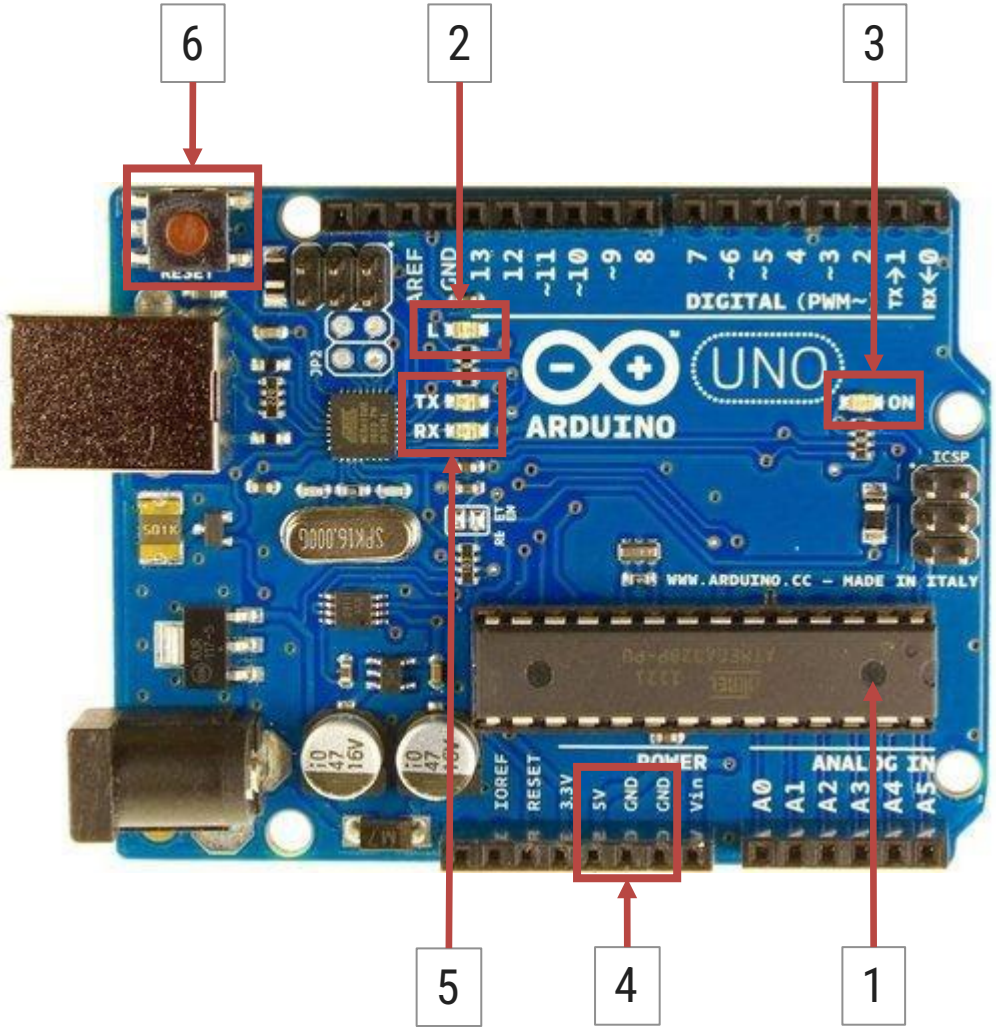






Introduction to Arduino Uno

- Atmega328p Microcontroller inside
- The operating voltage is 5V which is given by
  - USB socket or
  - External power socket
- DC Current for each input/output pin is 40 mA
- Digital input/output pins are 14 to interface digital input or output devices
- There are 6 Analog i/p pins used for interfacing analog input devices.
- What is Digital and Analog?
- Memory Components:
  - 32KB flash (program) memory
  - SRAM is 2 KB
  - EEPROM is 1 KB



Components in Arduino Uno

1. Atmega328p Microcontroller: The heart of the board.
2. On-board LED interfaced with pin 13.
3. Power LED: Indicates the status of Arduino whether it is powered or not.
4. GND and 5V pins: Used for providing +5V and ground to other components of circuits.
5. TX and RX LEDs: These LEDs indicate communication between Arduino and computer.
6. Reset button: Use to reset the ATmega328P microcontroller

Sensors and Their Interfacing

Section - 2

Definition of Sensor

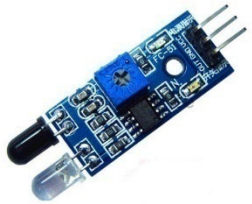
- **Sensor** is a device that detects or measures a physical property and records, indicates, or otherwise responds to it.
- A **sensor** is a device that measures **physical input** from its environment and converts it into data that can be **interpreted by either a human or a machine**.
- Normally, the sensors are input devices and there are two types of sensors.
  - **Analog Sensors**
  - **Digital Sensors**

List of Sensors

- The list of sensors which is to be covered in the chapter is as follows:

MQ-02/05 Gas Sensor

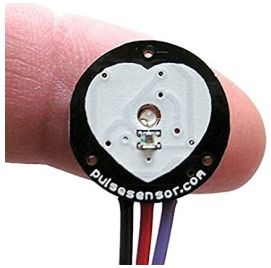
Obstacle Sensor



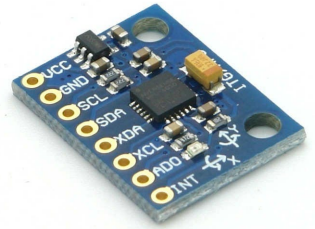
Ultrasonic Distance Sensor

LDR Sensor

Heartbeat Sensor

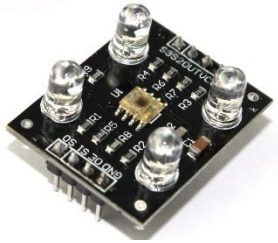


Gyro Sensor



GPS Sensor

Color Sensor



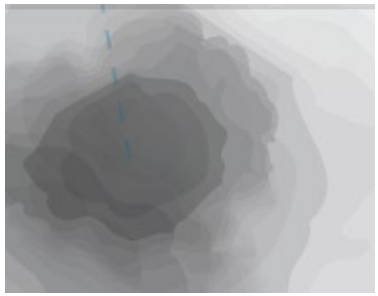
pH Sensor

MQ-02/05 Gas Sensor

- MQ -02 sensor is used to **detect smoke.**
- H2, LPG, CH4 alcohol and smoke can be detected by MQ-02.
- MQ-05 **is not sensitive to smoke**, hence less used in the application.
- Pinout of MQ-02 sensor module are
  - Vcc – Connected to 5V
  - Gnd – Connected to ground
  - D0 – Digital output pin
  - A0 – Analog output pin

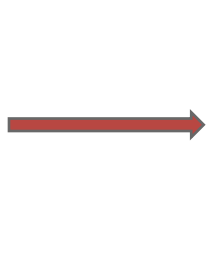
MQ-02/05 Gas Sensor – Working principle

- When any flammable gas passes through the coil of the sensor, the coil burns and **internal resistance decreases.**
- This results in an **increased voltage** across it. Hence we get variable voltage at A0 pin of MQ-05 gas sensor.
- If gas present -> voltage is high.
- If gas is not present -> voltage is low.



Smoke/Gas

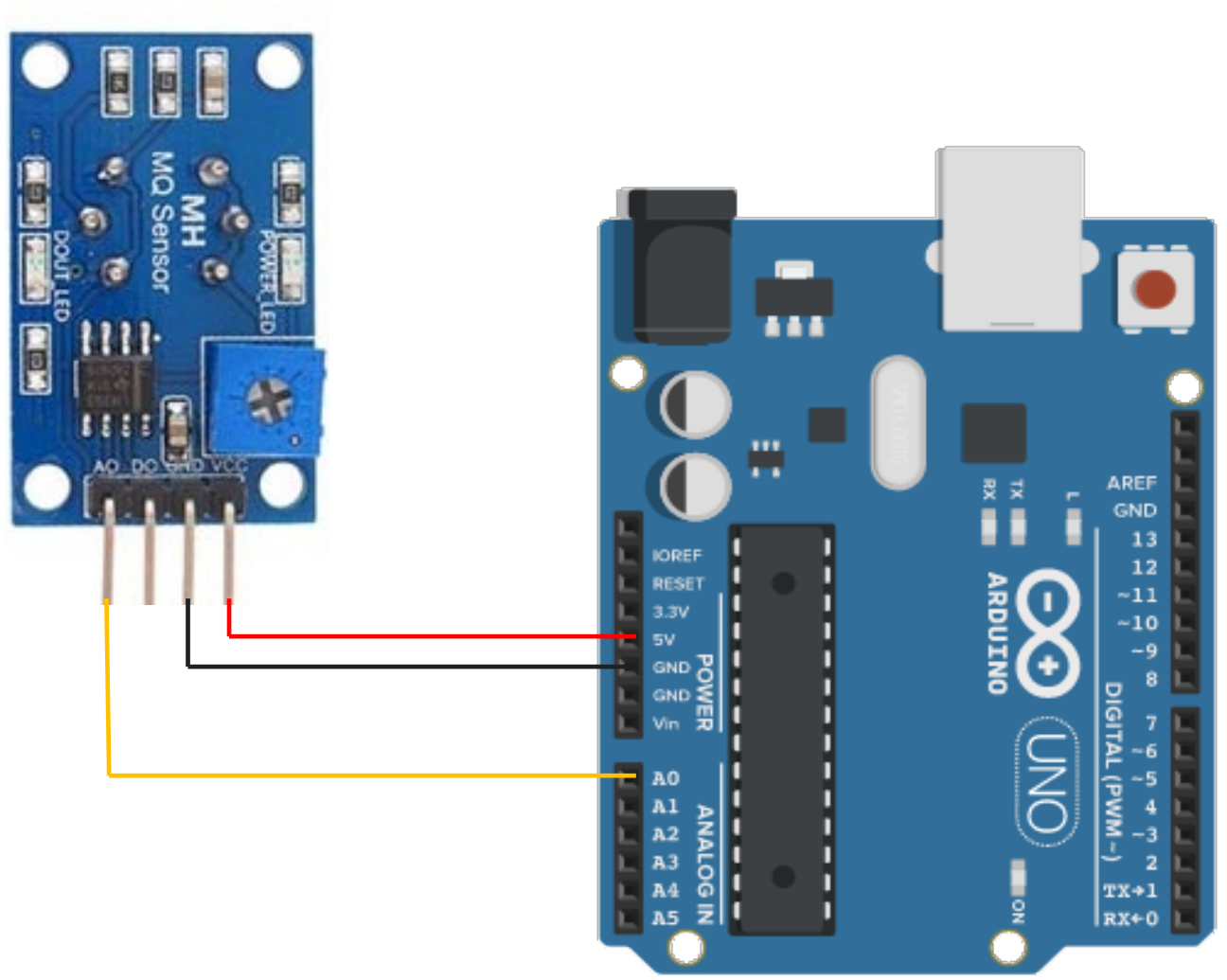
MQ-05 Gas Sensor



Variable Voltage

MQ-02/05 Gas Sensor – Interfacing with Arduino

- The interfacing of MQ-02 with Arduino Uno is as shown in the figure.
- Here, Vcc and Gnd pins of MQ-02 are connected to 5V and GND of Arduino board.
- We want to take the input from a Gas sensor in **an analog** format so pin A0 is connected to one of the analog pins of Arduino i.e. A0.



MQ-02/05 Gas Sensor – Code explanation

- The code in Arduino for MQ-02 gas sensor can be written as following

gas_sensor.ino

```
1  int gas_level;
2  void setup() {
3    pinMode(A0, INPUT);
4    Serial.begin(9600);
5  }
6
7  void loop() {
8    gas_level = analogRead(A0);
9    Serial.print("Gas Level : ");
10   Serial.println(gas_level);
11   delay(500);
12 }
```

Initialize the serial communication between Arduino and computer with baud rate 9600.

Reads the analog value from specified pin and stores it in the mentioned variable.

Prints data on the serial port in ASCII text given in double quotes.

Prints the data of specified variable on the serial port and also prints new line at the end.

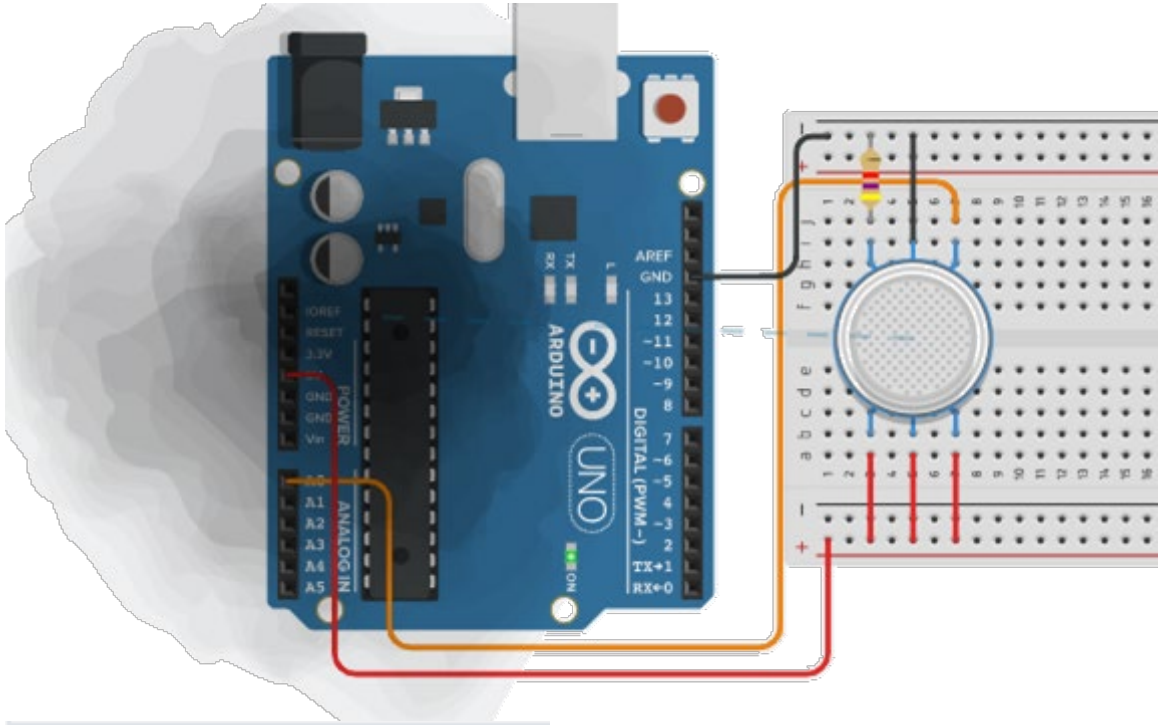
Functions in Arduino IDE

- Serial.begin( ): It is used to start serial communication between Arduino board and other device. Normally, it is connected with computer for monitoring the data. Serial communication uses pin 0 and 1 also working as Rx and Tx respectively.
  - Syntax : Serial.begin(baud_rate)
  - Parameters :
    - baud_rate: It defines the speed of data transfer rate between Arduino and other device. It is compulsory to set this baud rate same at both the side for proper transfer of data.
- Serial.print( ) : Print the data from Arduino to other device.
  - Syntax : Serial.print(“text”)
  - Parameters :
    - The text which is to be printed by Arduino board is written in double quote so as to convert it in its ASCII values.
- Serial.println( ) : Print the data with newline from Arduino to other device.
  - Syntax : Serial.println(“text”)
  - Parameters :
    - The text which is to be printed by Arduino board is written in double quote so as to convert it in its ASCII values.

Functions in Arduino IDE (Cont.)

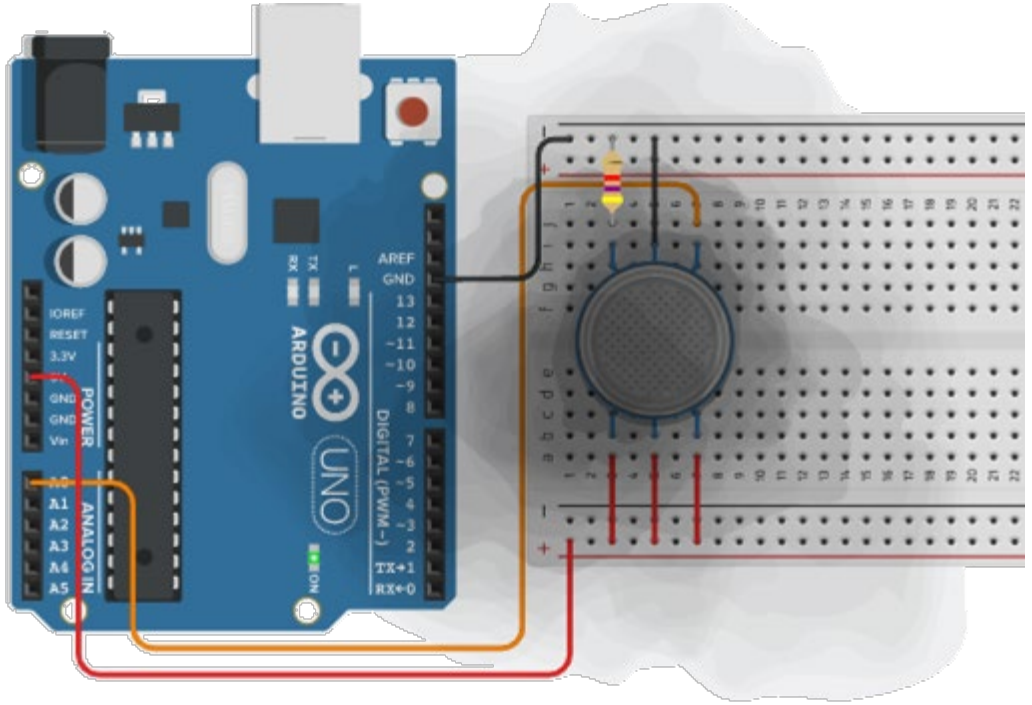
- analogRead( ): The syntax reads analog value from specified pin and converts it into 1024 levels or in 0 to 1023 range. The values is stored into specified variable in the code statement. The value is converted into 1024 levels because Arduino has 10 bit built in A-D converter unit.
  - Syntax : analogRead(pin)
  - Parameters :
    - pin: The Arduino analog pin number.

MQ-02/05 Gas Sensor – Output



Serial Monitor

```
Gas Level : 306
Gas Level : 306
Gas Level : 306
Gas Level : 306
Gas Level : 306
Gas Level : 306
```

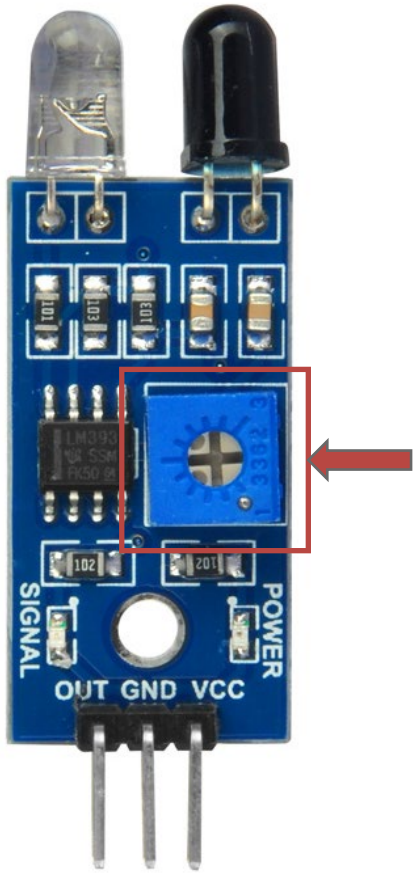


Serial Monitor

```
Gas Level : 745
Gas Level : 745
Gas Level : 745
Gas Level : 745
Gas Level : 745
Gas Level : 745
```

Obstacle Sensor

- Obstacle sensor is used for **detection of obstacle**.
- It is a digital sensor, hence gives binary output '1' or '0'.
- The **range** for detection of obstacle can be changed by **potentiometer** given.
- Pinout of obstacle sensor module are
  - Vcc – Connected to 5V
  - Gnd – Connected to ground
  - Out/D0 – Digital output pin



Obstacle Sensor – Working principle

- IR emitter transmits IR signals and IR receiver receives those signals from reflection.
- If the obstacle is present then the transmitted signals are **reflected** back by obstacle and if they have amplitude greater then threshold the output signal will be **'0'**.
- If the obstacle is not present then the transmitted signals are **not reflected** and the output signal will be **'1'**.

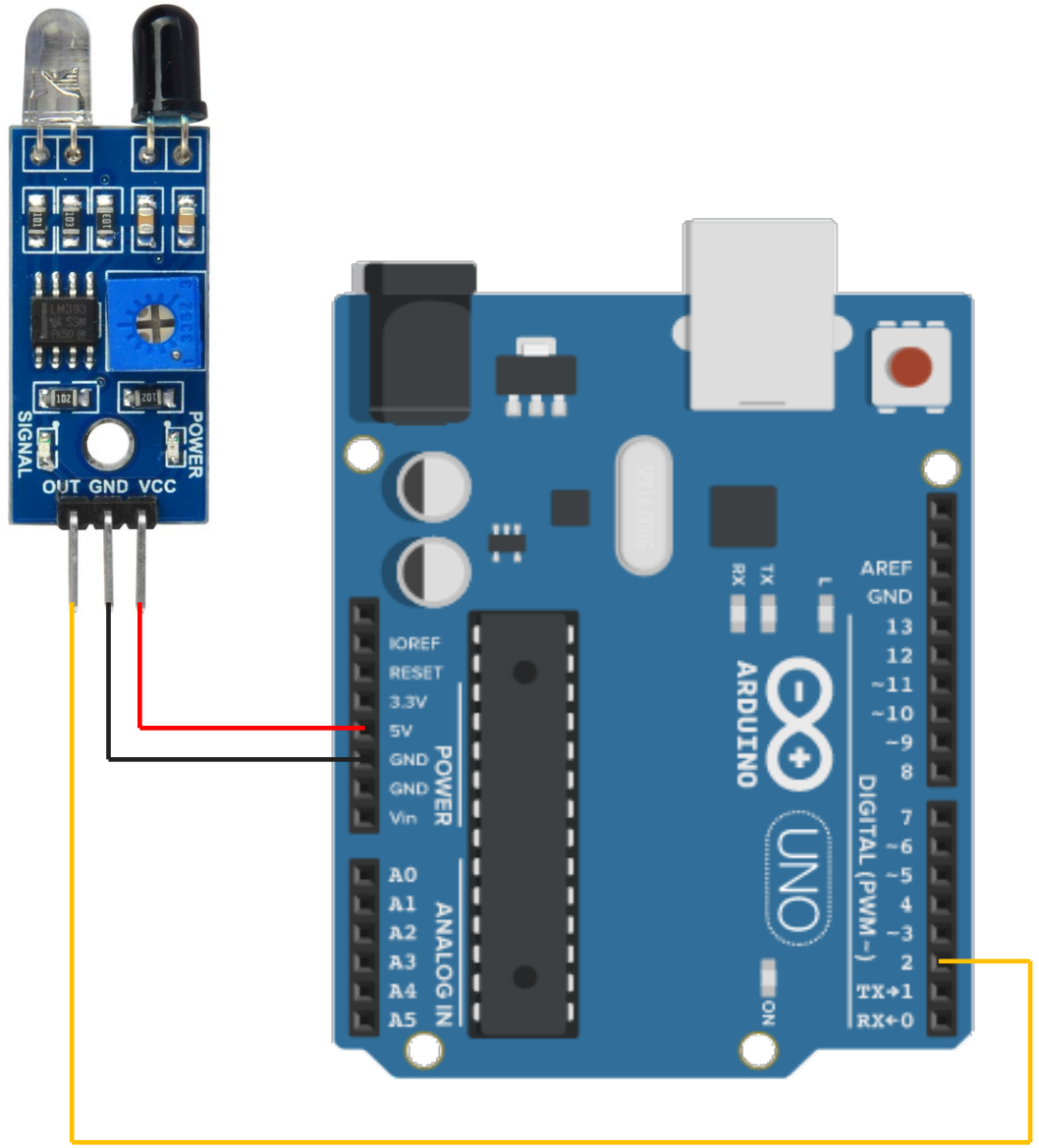
Obstacle in the way

Obstacle IR Sensor

Digital Output

Obstacle Sensor – Interfacing with Arduino

- The interfacing of obstacle sensor with Arduino Uno is as shown in figure.
- Here, Vcc and Gnd pins of obstacle sensor is connected to 5V and Gnd of Arduino board.
- The output of obstacle sensor is in **digital** form. So it is connected to digital pin 2 of Arduino.



Obstacle Sensor – Code explanation

- The code in Arduino for Obstacle sensor can be written as following:

obstacle-detect.ino

```
1  int obstacle_pres=0;
2  void setup() {
3    pinMode(2, INPUT);
4    pinMode(13, OUTPUT);
5    Serial.begin(9600);
6  }
7
8  void loop() {
9    obstacle_pres = digitalRead(2);
10   if (obstacle_pres == LOW)
11   {
12      Serial.print("Obstacle is present");
13      digitalWrite(13,HIGH);
14   }
```

Reads the digital data from specified pin and stores it into given variable.

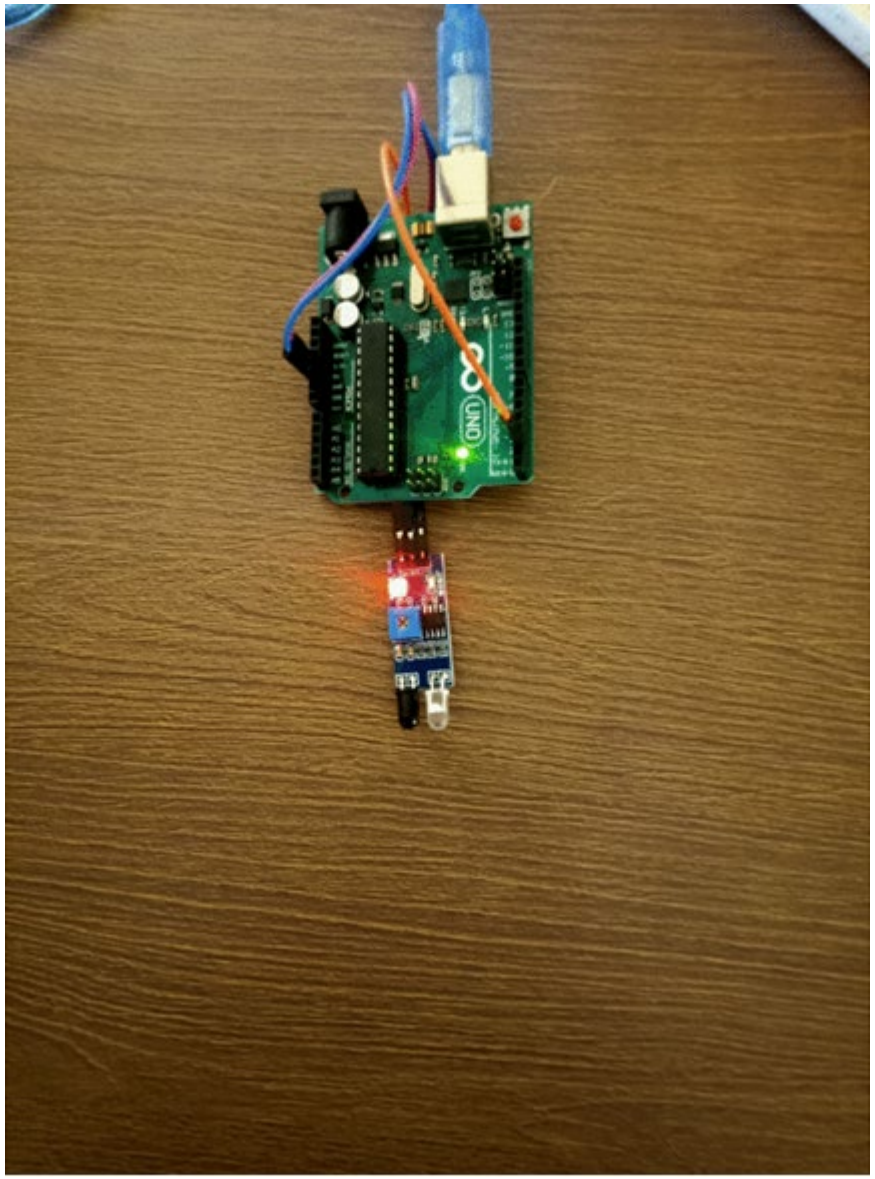
Obstacle Sensor – Code explanation (Cont.)

obstacle-detect.ino

```
13    else
14    {
15       digitalWrite(13,LOW);
16    }
17    delay(1000);
18 }
```

- digitalRead( ) : Reads digital data from specified pin and stores it into given variable.
  - Syntax : digitalRead(pin)
  - Parameters :
    - Pin : The Arduino digital pin number.

Obstacle Sensor – Output

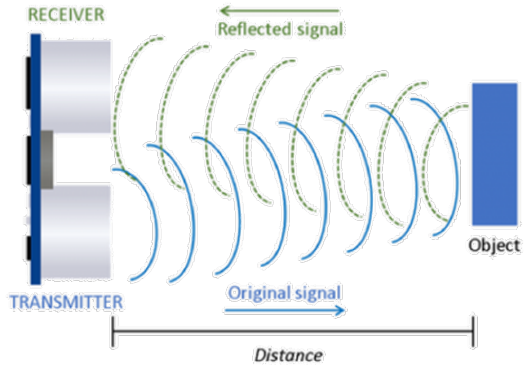


HC-SR04 Ultrasonic Sound Sensor

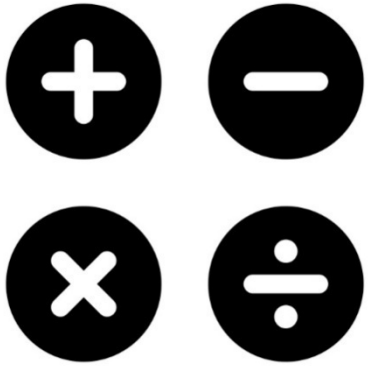
- Ultrasonic Sound Sensor is used to measure the **distance of an object.**
- The sensor can only measure the distance of object. It can not identify the type of object.
- It is also known as Ultrasonic distance sensor.
- Pinout of HC-SR04 module are
  - Vcc – Connected to 5V
  - Gnd – Connected to ground
  - Trigger – Generates the transmit pulse
  - Echo – Receives echo from obstacle

HC-SR04 Ultrasonic Sound Sensor – Working principle

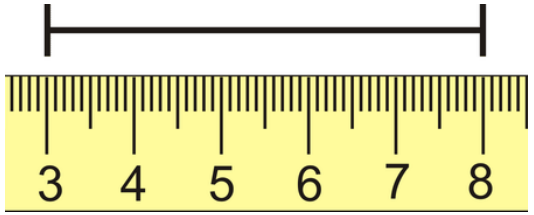
- The emitter transmits ultrasonic sound waves which are reflected by near by objects.
- The reflected pulse is received by sensor.
- Some mathematical operations are performed to obtain the distance value.
- The distance value is converted into desired unit.



Ultrasonic waves



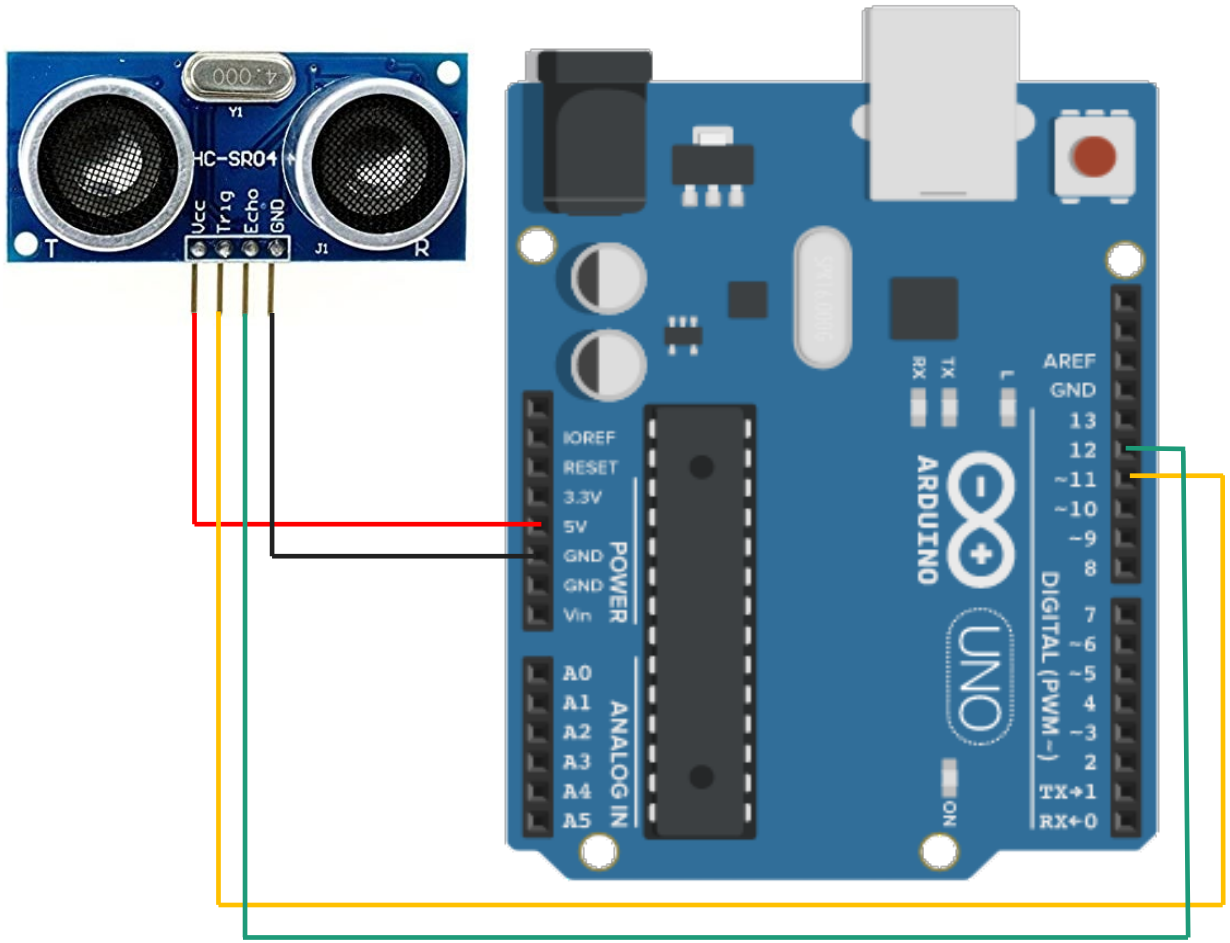
Mathematical operations



Measurement of distance

HC-SR04 Ultrasonic Sound Sensor– Interfacing with Arduino

- The interfacing of HC-SR04 sensor with Arduino Uno is as shown in figure.
- Here, Vcc and Gnd pins of HC-SR04 sensor are connected to 5V and Gnd of Arduino board.
- The trigger and echo pin is connected to (any) digital pins of Arduino board.
- In this case, trigger is connected to 11 and echo is connected to 12.



HC-SR04 Ultrasonic Sound Sensor – Code explanation

- The code in Arduino for HC-SR04 Ultrasonic Sound Sensor can be written as following

Distance-measure.ino

```
1  int trigPin = 11;     // Trigger
2  int echoPin = 12;     // Echo
3  long duration, cm, inches;
4
5  void setup() {
6    Serial.begin (9600);
7    pinMode(trigPin, OUTPUT);
8    pinMode(echoPin, INPUT);
9  }
10 void loop() {
11   digitalWrite(trigPin, LOW);
12   delayMicroseconds(5);
13   digitalWrite(trigPin, HIGH);
14   delayMicroseconds(10);
15   digitalWrite(trigPin, LOW);
```

Generates the delay of specified microseconds.

HC-SR04 Ultrasonic Sound Sensor – Code explanation (Cont.)

Distance-measure.ino

```
16    pinMode(echoPin, INPUT);
17    duration = pulseIn(echoPin, HIGH);
18
19    // Convert the time into a distance
20    cm = (duration/2) / 29.1;
21    inches = (duration/2) / 74;
22    Serial.print(inches);
23    Serial.print("in, ");
24    Serial.print(cm);
25    Serial.print("cm");
26    Serial.println();
27
28    delay(250);
29  }
```

Reads a HIGH pulse on specified pin and returns the value of time in microsecond for transition from LOW to HIGH.

Distance(m) = Speed (m/s)x Time (s)

But, Time of pulse we receive is in µs and distance we want to find is in cm.
Distance(cm) = Speed (cm/µs) x Time (µs)

Speed of sound is 343m/s which is converted into 0.0343 cm/µs.
So either we have to multiply 0.0343 or divide 29.1

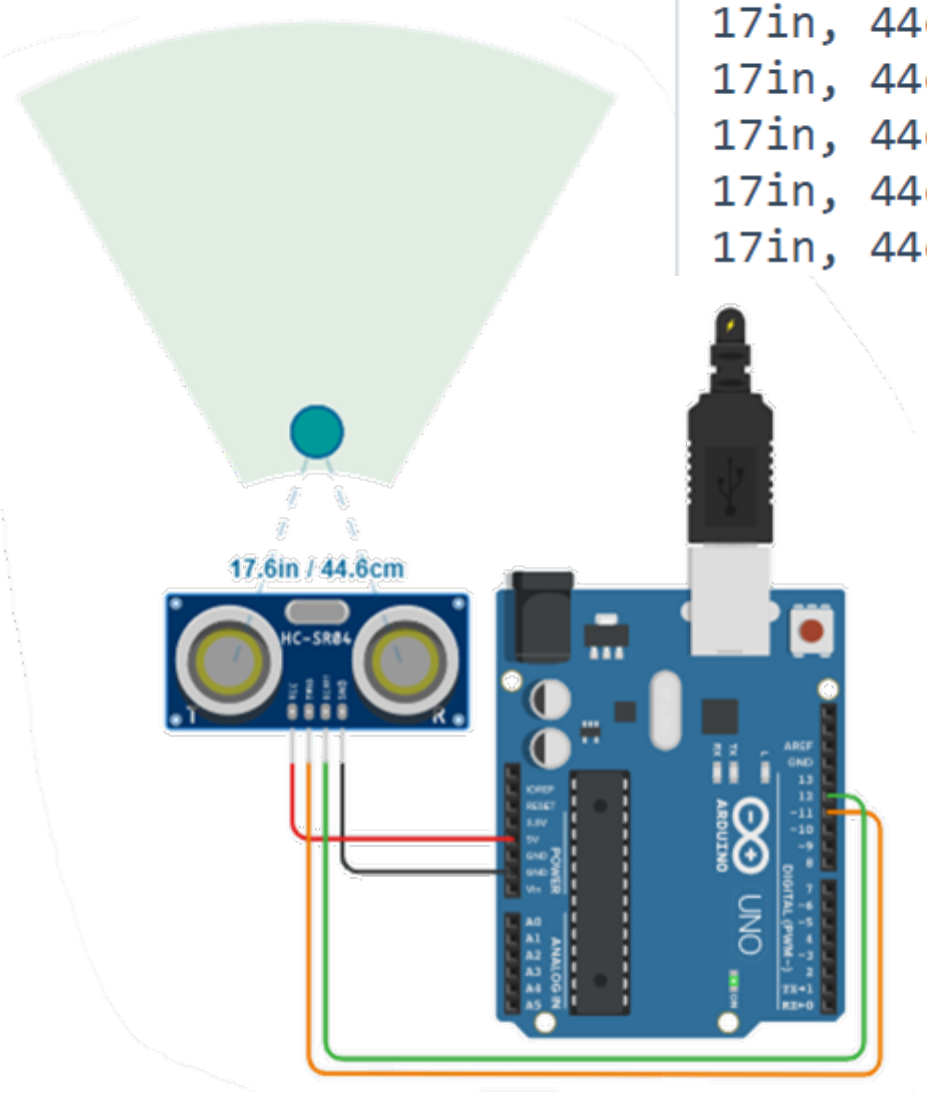
Functions in Arduino IDE

- delayMicroseconds( ): It pauses the program for amount of time in microseconds specified as the parameter.
  - Syntax : delayMicroseconds(μs)
  - Parameters :
    - μs : The number of microseconds to pause.
- pulseIn( ) : Reads a pulse HIGH or LOW from specified pin and wait for the time to go the pulse from HIGH to LOW or LOW to HIGH. It returns the time required to transit the pulse in variable.
  - Syntax : pulseIn(pin, value)
  - Parameters :
    - Pin: The number of the Arduino pin on which you want to read the pulse.
    - Value: The type of pulse that you want to read.

HC-SR04 Ultrasonic Sound Sensor – Output

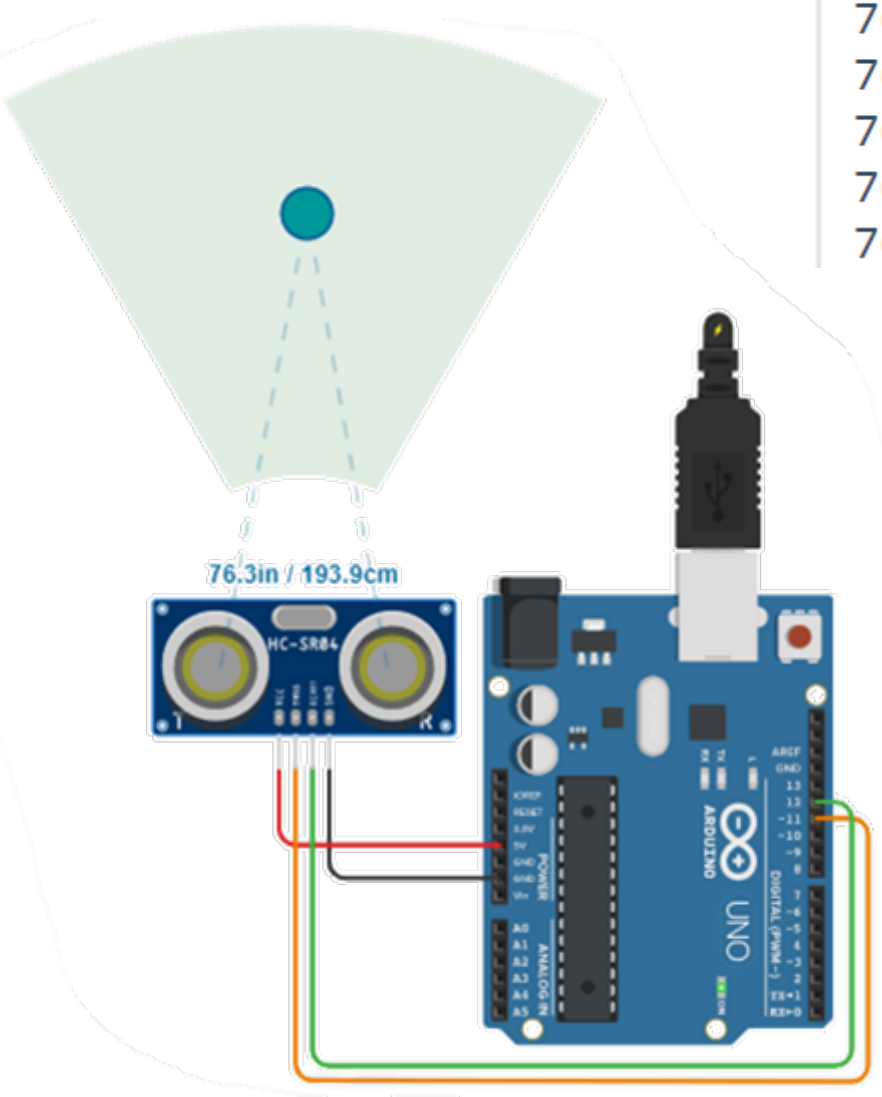
Serial Monitor

```
17in, 44cm
17in, 44cm
17in, 44cm
17in, 44cm
17in, 44cm
```



Serial Monitor

```
76in, 193cm
76in, 193cm
76in, 193cm
76in, 193cm
76in, 193cm
```

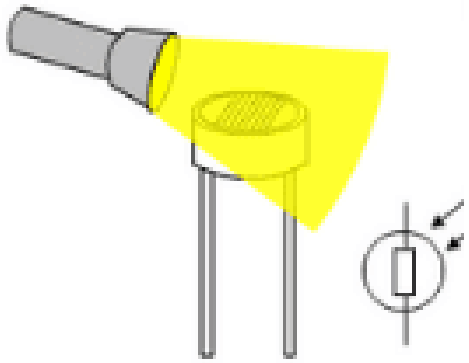


LDR Sensor

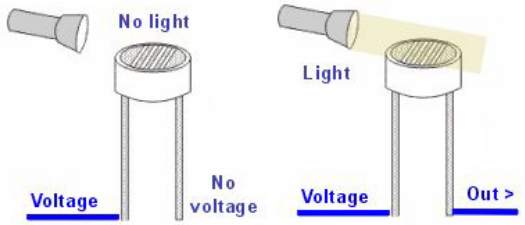
- LDR is known as Light Dependent Resistor.
- The **internal resistance** of LDR changes with respect to light intensity.
- Normally LDR works with voltage divider network.
- The LDR is used in the application where the task is performed with light intensity as input.

LDR Sensor – Working principle

- The LDR works on principle of Ohm's law.
- As the light intensity on LDR is higher, the resistance of LDR decreases.
- The decreased resistance will drop more voltage across LDR according to Ohm's Law.
- The presence of light can be identified by the value of voltage across LDR.



Light Intensity to be measured

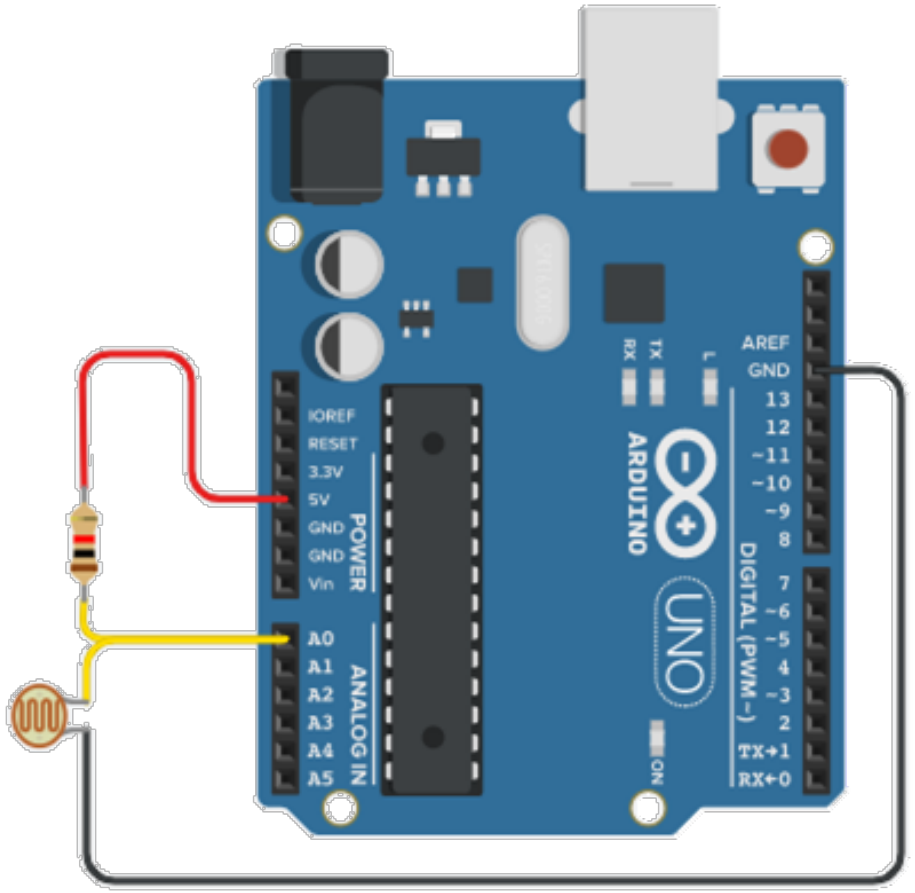


Output voltage measurement

Calculate the intensity

LDR Sensor– Interfacing with Arduino

- The interfacing of LDR sensor with Arduino Uno is as shown in figure.
- Here, one terminal of 10kΩ resistor is connected to 5V of Arduino and second terminal is connected to A0 pin.
- One terminal of LDR is connected to A0 pin of Arduino and other is connected to Gnd.
- This creates a voltage divider network between fixed resistor and LDR.
- The voltage at A0 pin remains near to 5V when LDR is in dark (high resistance).
- The voltage at A0 pin remains near to 0V when LDR is in Bright light (low resistance).



LDR Sensor – Code explanation

- The code in Arduino for LDR Sensor can be written as following

Light-intensity-measure.ino

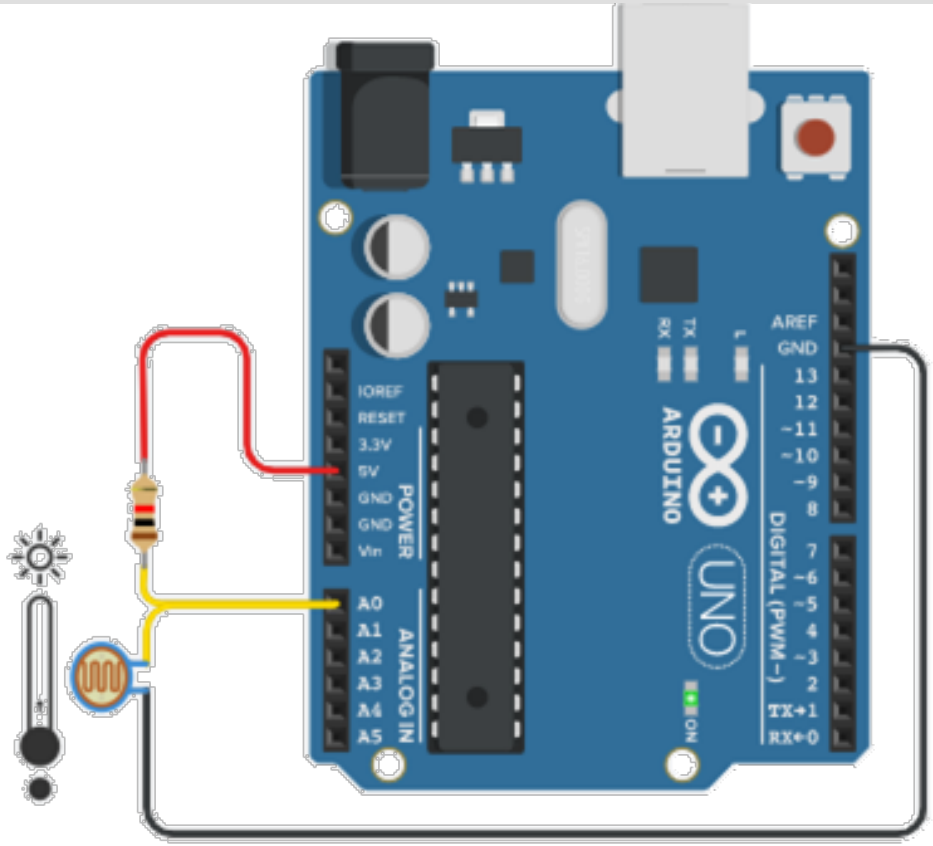
```
1  void setup()
2  {
3    pinMode(A0, INPUT);
4    Serial.begin(9600);
5  }
6
7  void loop()
8  {
9    int light_intensity = analogRead(A0);
10   if (light_intensity > 800)
11   {
12     Serial.println(“Darker");
13   }
```

LDR Sensor – Code explanation (Cont.)

Light-intensity-measure.ino

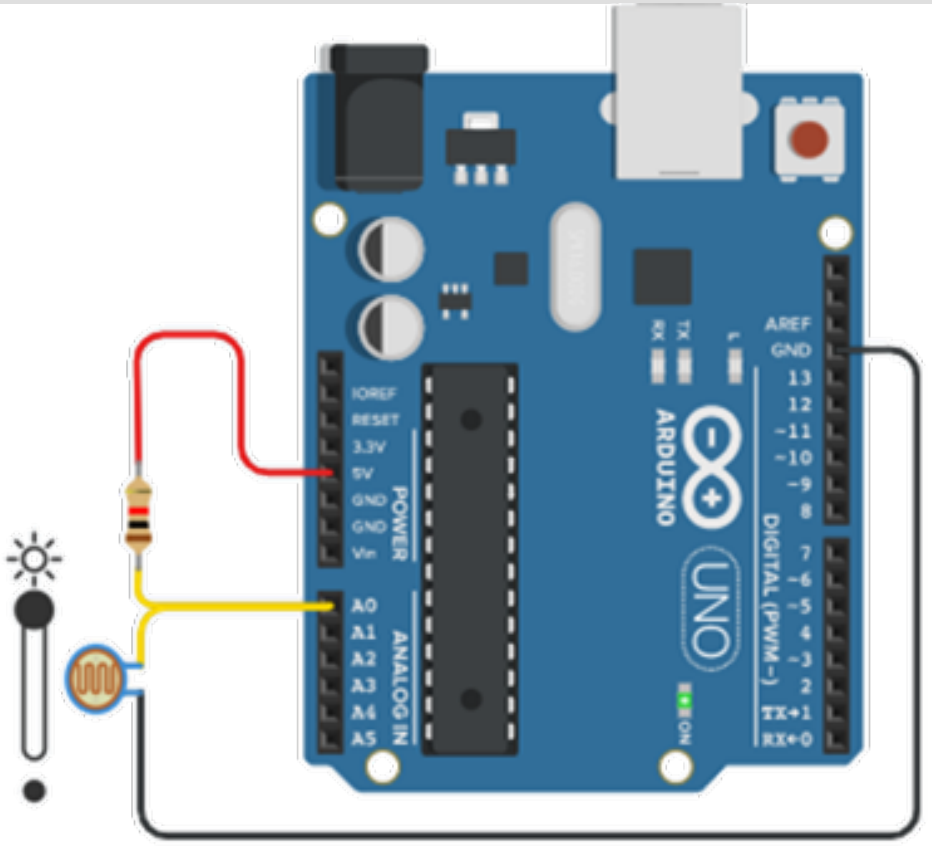
```
16    else if (light_intensity < 500)
17    {
18      Serial.println(“Too Bright");
19    }
20    delay(1000);
21  }
```

LDR Sensor – Output



Serial Monitor

```
Darker
Darker
Darker
Darker
```



Serial Monitor

```
Too Bight
Too Bight
Too Bight
Too Bight
```

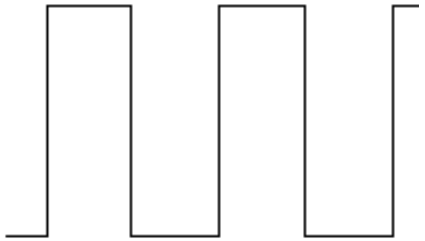
Heartbeat Sensor

- Heartbeat sensor is having typical application in health care domain.
- This is mainly used in wearable devices to monitor the heart beat / heart rate of the person.
- The sensor is inexpensive and generates square waves for each pulse. The averaging of pulses gives analog voltage according to the heartbeat.
- Pinout of heartbeat sensor module are
  - Vcc – Connected to 5V
  - Gnd – Connected to ground
  - A0 – Analog output pin

Heartbeat Sensor – Working principle

- The sensor senses the heartbeat as per blood circulation in the body.
- It generates the output pulse signal according to the rate at which heart beats.
- The output pulses are considered as PWM waves. Hence they are converted into analog signals.
- This analog signal is converted to digital which gives the output beat rate.

Heartbeat input

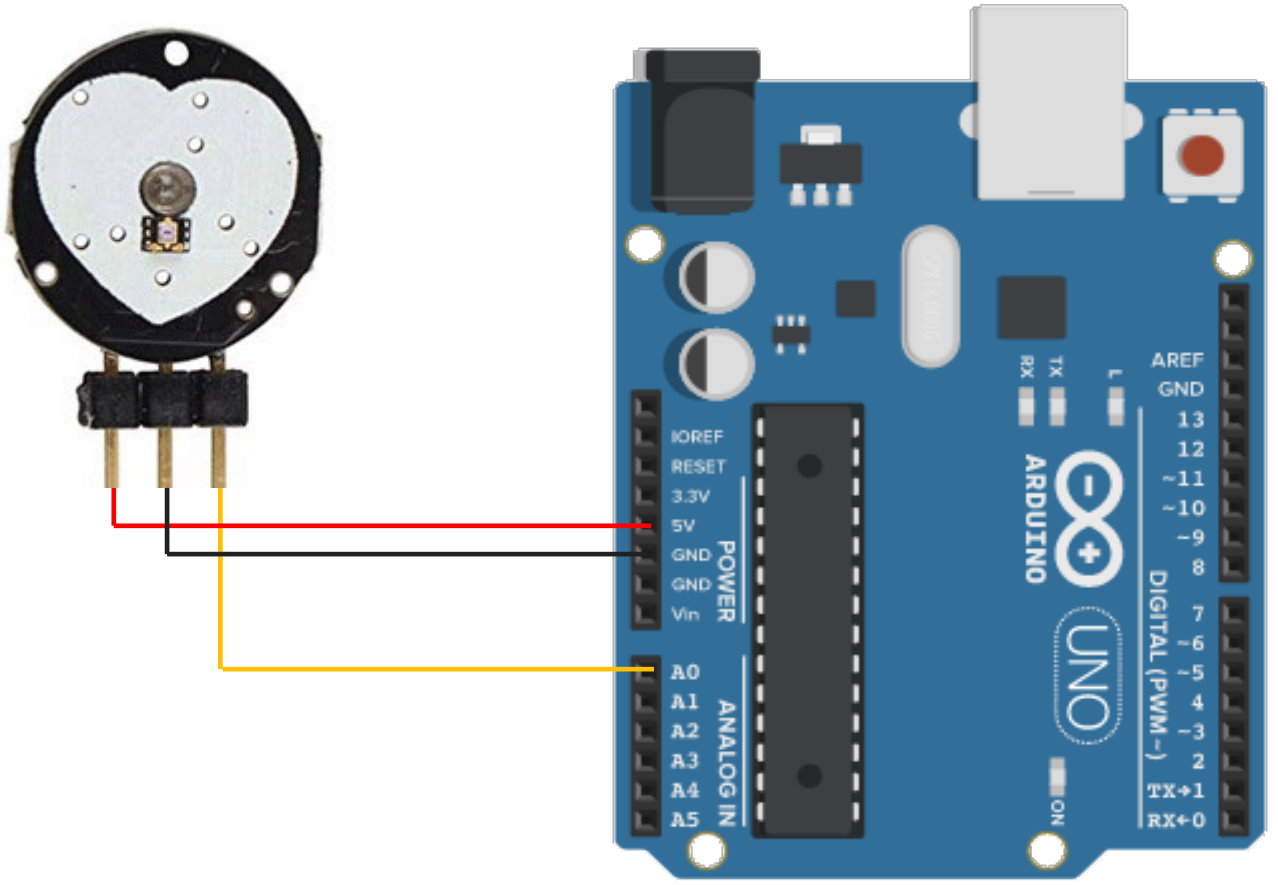


Generation of Digital Pulse as output

Calculates the BPM

Heartbeat Sensor – Interfacing with Arduino

- The interfacing of Heartbeat sensor with Arduino Uno is as shown in the figure.
- Here, Vcc and Gnd pins of heartbeat sensor are connected to 5V and GND of Arduino board.
- We want to take the input from the sensor in **an analog** format so the signal pin of the sensor is connected to one of the analog pins of Arduino i.e. A0.



Heartbeat Sensor – Code explanation

- The code in Arduino for Heartbeat sensor can be written as following:

Heartbeat_sensor.ino

```
int UpperThreshold = 518;
int LowerThreshold = 490;
int reading = 0;
float BPM = 0.0;
bool IgnoreReading = false;
bool FirstPulseDetected = false;
unsigned long FirstPulseTime = 0;
unsigned long SecondPulseTime = 0;
unsigned long PulseInterval = 0;

void setup(){
  Serial.begin(9600);
}
```

Heartbeat Sensor – Code explanation (Cont.)

Heartbeat_sensor.ino

```
14 void loop(){
15   reading = analogRead(0);
16   if(reading > UpperThreshold
17        && IgnoreReading == false){
18     if(FirstPulseDetected == false){
19       FirstPulseTime = millis();
20       FirstPulseDetected = true;
21     }
22     else{
23       SecondPulseTime = millis();
24       PulseInterval = SecondPulseTime -
25                              FirstPulseTime;
26       FirstPulseTime = SecondPulseTime;
27     }
28     IgnoreReading = true;
29   }
```

millis( ) function gives the time in millisecond from where the Arduino board powered or reset.

Heartbeat Sensor – Code explanation (Cont.)

Heartbeat_sensor.ino

```
30    if(reading < LowerThreshold){
31      IgnoreReading = false;
32    }
33
34    BPM = (1.0/PulseInterval) * 60.0 * 1000;
35    Serial.print(BPM);
36    Serial.println(" BPM");
37    Serial.flush();
38  }
```

PulseInterval is in milliseconds.

$\therefore$ Rate = 1/Time

$$\therefore \text{Rate} = \frac{1}{\text{PulseInterval(ms)}}$$

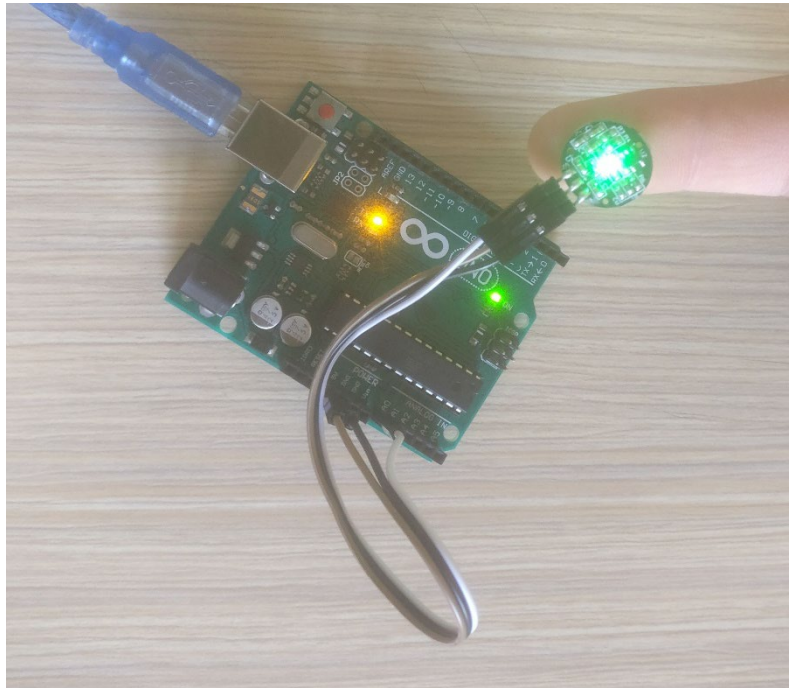
$$\therefore \text{Rate} = \frac{1}{\text{PulseInterval}\,(s)/1000}$$

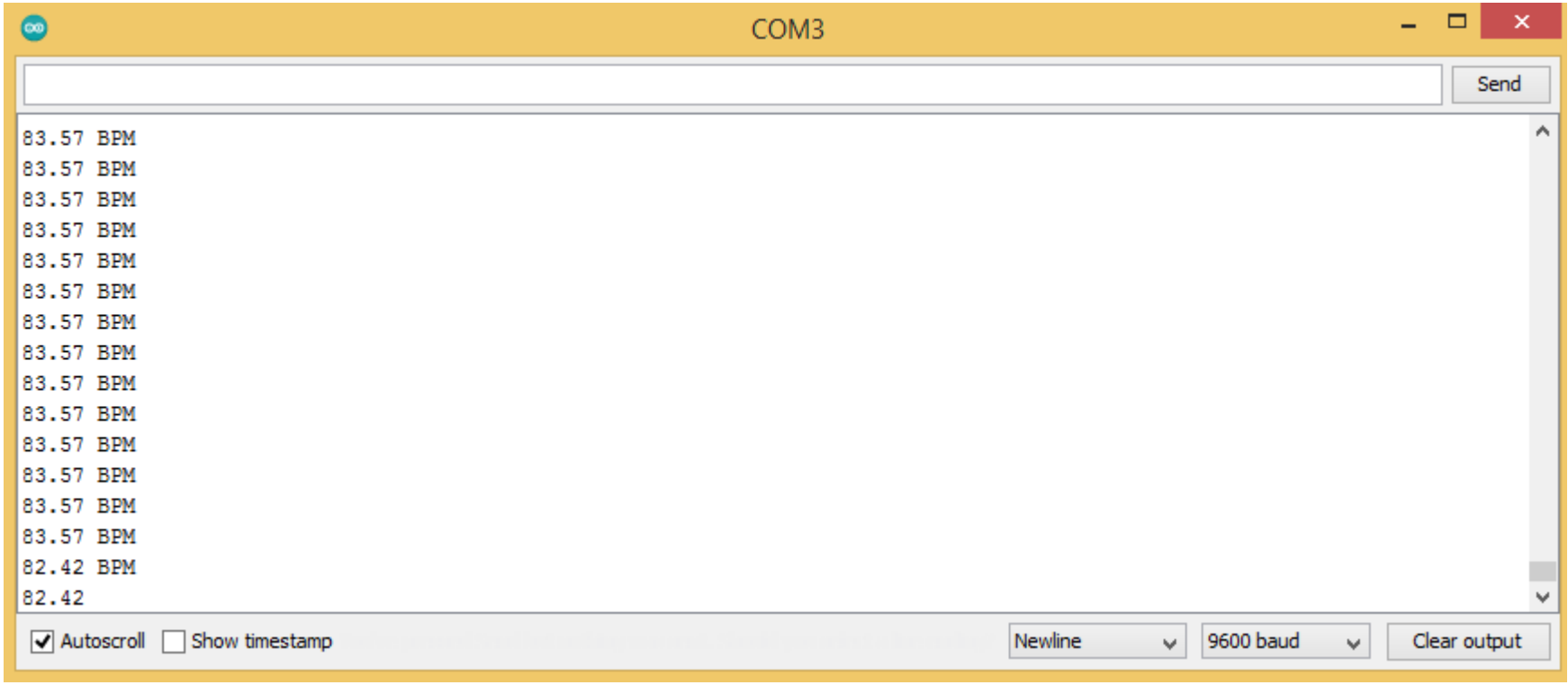
$$\therefore \text{Rate} = \frac{1}{\text{PulseInterval(s)}} * 1000$$

$$\therefore \text{Rate in min} = \frac{1}{\text{PulseInterval}} * 1000 * 60$$

- millis( ) : Returns the number of milliseconds passed since the Arduino started running the current program.
  - Syntax : millis( )

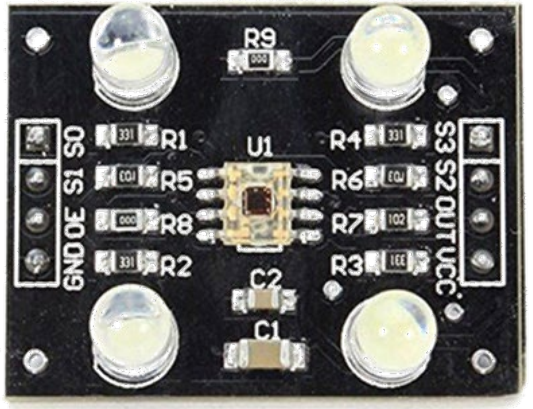
Heartbeat Sensor – Output





Colour Sensor

- Colour sensor is used to detect the RGB colour coordinates of a particular colour.
- The colour sensor module has TSC3200 IC that converts colour to frequency by enabling a particular colour diode turn by turn.
- The colour sensor is mainly used in the application where the objects are identified by their colour.
- Pinout of colour sensor module are
  - Vcc – Connected to 5V
  - Gnd – Connected to ground
  - S0, S1 – Used for output frequency scaling
  - S2, S3 – Type of photodiode selected
  - (OE)' – Enable for output
  - OUT – Output frequency (f_o)



Colour Sensor (Cont.)

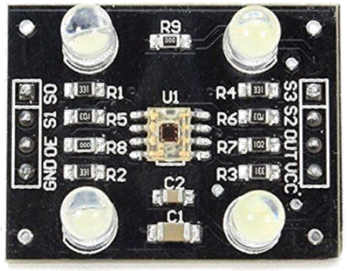
- The combination of S0 and S1 are used for output frequency scaling.
- The different microcontrollers have different counter functionality and limitations. Hence, we need to scale the frequency according to microcontroller used.
- The percentage of output scaling for different combinations of S0 and S1 are shown in the table.
- The S3 and S4 pins are used for photodiode selection. The photodiodes are associated with different colour filters.
- The table shows combinations of S3 and S4 for different photodiodes.

S0	S1	Output Frequency Scaling (f_0)	Typical full-scale Frequency
L	L	Power Down	–––
L	H	2%	10 – 12 KHz
H	L	20%	100 – 120 KHz
H	H	100%	500 – 600 KHz

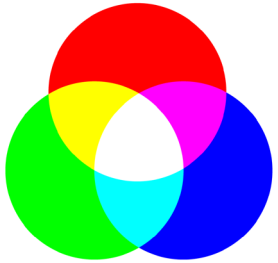
S3	S4	Photo Diode Type
L	L	Red
L	H	Blue
H	L	Clear(No Filter)
H	H	Green

Colour Sensor – Working principle

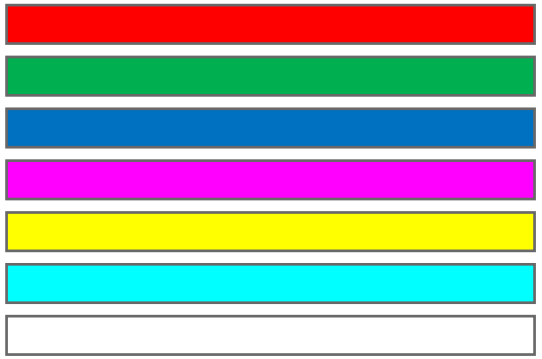
- The sensor works by imparting a bright light on an object and recording the reflected colour by the object.
- The selected photodiode for colour of red, green and blue converts the amount of light to current.
- These RGB values are further processed to identify the exact colour combination.



Input to photodiode of colour sensor



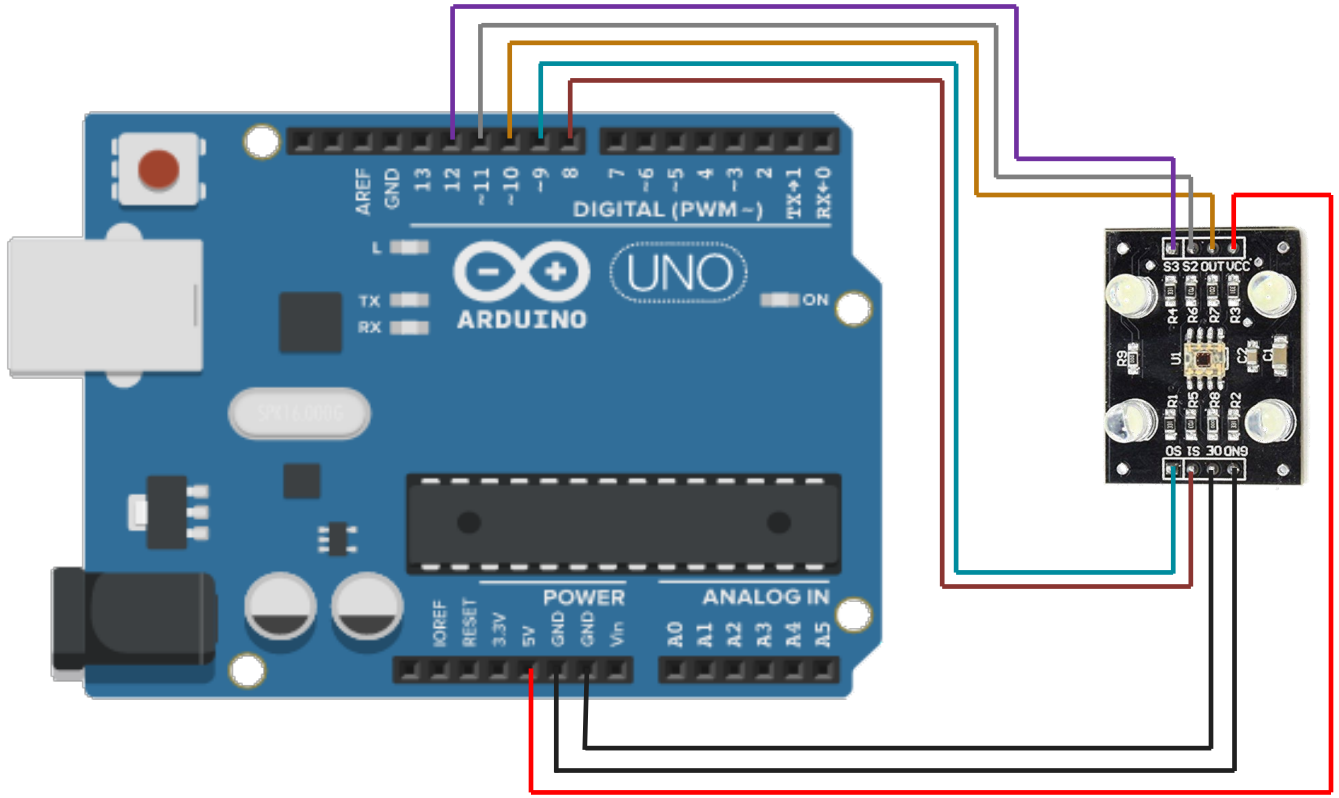
Output RGB colour values



Calculation of actual colour of object

Colour Sensor – Interfacing with Arduino

- The interfacing of Colour Sensor with Arduino Uno is as shown in figure.
- Here, Vcc and Gnd pins of Colour Sensor are connected to 5V and Gnd of Arduino board.
- As we need to enable the sensor OE' is connected to GND.
- All selection pins S0, S1 S2 and S3 are connected to digital pins of Arduino that is 9,8,11 and 12 respectively.



- The output of colour sensor are taken as a pulse count. So OUT pin is also connected to digital pin. In our case it is connected to pin 10.

Colour Sensor – Code explanation

- The code in Arduino for Colour sensor can be written as following:

Colour_sensor.ino

```
#define s0 9
#define s1 8
#define s2 11
#define s3 12
#define out 10

int frequency=0;

void setup()
{
   pinMode(s0,OUTPUT);
   pinMode(s1,OUTPUT);
   pinMode(s2,OUTPUT);
   pinMode(s3,OUTPUT);
   pinMode(out,INPUT);
```

Colour Sensor – Code explanation (Cont.)

Colour_sensor.ino

```
16    Serial.begin(9600);
17    digitalWrite(s0,HIGH);
18    digitalWrite(s1,HIGH);
19 }
20 void loop()
21 {
22    digitalWrite(s2,HIGH);
23    digitalWrite(s3,HIGH);
24    frequency = pulseIn(out,LOW);
25    frequency = map(frequency, 30,90,255,0);
26    Serial.print("G = ");
27    Serial.print(frequency);
28    Serial.print("\t");
29    delay(100);
30
```

This syntax maps the value of given number or variable from its defined range to desired range.

Colour Sensor – Code explanation (Cont.)

Colour_sensor.ino

```
31      digitalWrite(s2,LOW);
32      digitalWrite(s3,HIGH);
33      frequency = pulseIn(out,LOW);
34      frequency = map(frequency,25,70,255,0);
35      Serial.print("B = ");
36      Serial.print(frequency);
37      Serial.print("\t");
38      delay(100);
39      digitalWrite(s2,LOW);
40      digitalWrite(s3,HIGH);
41      frequency = pulseIn(out,LOW);
42      frequency = map(frequency,10,56,255,0);
43      Serial.print("R = ");
44      Serial.print(frequency);
45      Serial.print("\t");
46      delay(100);
47  }
```

Functions in Arduino IDE

- map( ): The syntax maps a number from one range to another.
  - Syntax : map(value, fromLow, fromHigh, toLow, toHigh)
  - Parameters :
    - value: the number or variable to map.
    - fromLow: the lower bound of the value’s current range.
    - fromHigh: the upper bound of the value’s current range.
    - toLow: the lower bound of the value’s target range.
    - toHigh: the upper bound of the value’s target range.

GPS Sensor

- GPS sensor is used to get the location data of the place where sensor is situated.
- The data from sensor is acquired in National Marine Electronics Association (NEMA) format. The format is used by marine department for communication.
- There are multiple variants of GPS sensors available in the market. We can select based on our requirements.
- Here, the given sensor is GY-GPS6MV2 as shown in the image.
- Pinout of GPS sensor module are
  - Vcc – Connected to 5V
  - Gnd – Connected to ground
  - Tx – Serial Data Transmit
  - Rx – Serial Data Receive

GPS Sensor – Working principle

- The sensor is placed in the system whose location is to be tracked.
- The GPS device communicates with GPS satellite and the satellite returns its current location in form of latitude and longitude.
- The sensor transmits the data to the microcontroller in a predefined format from which we can extract the tracked location.

GPS device with antenna for communication

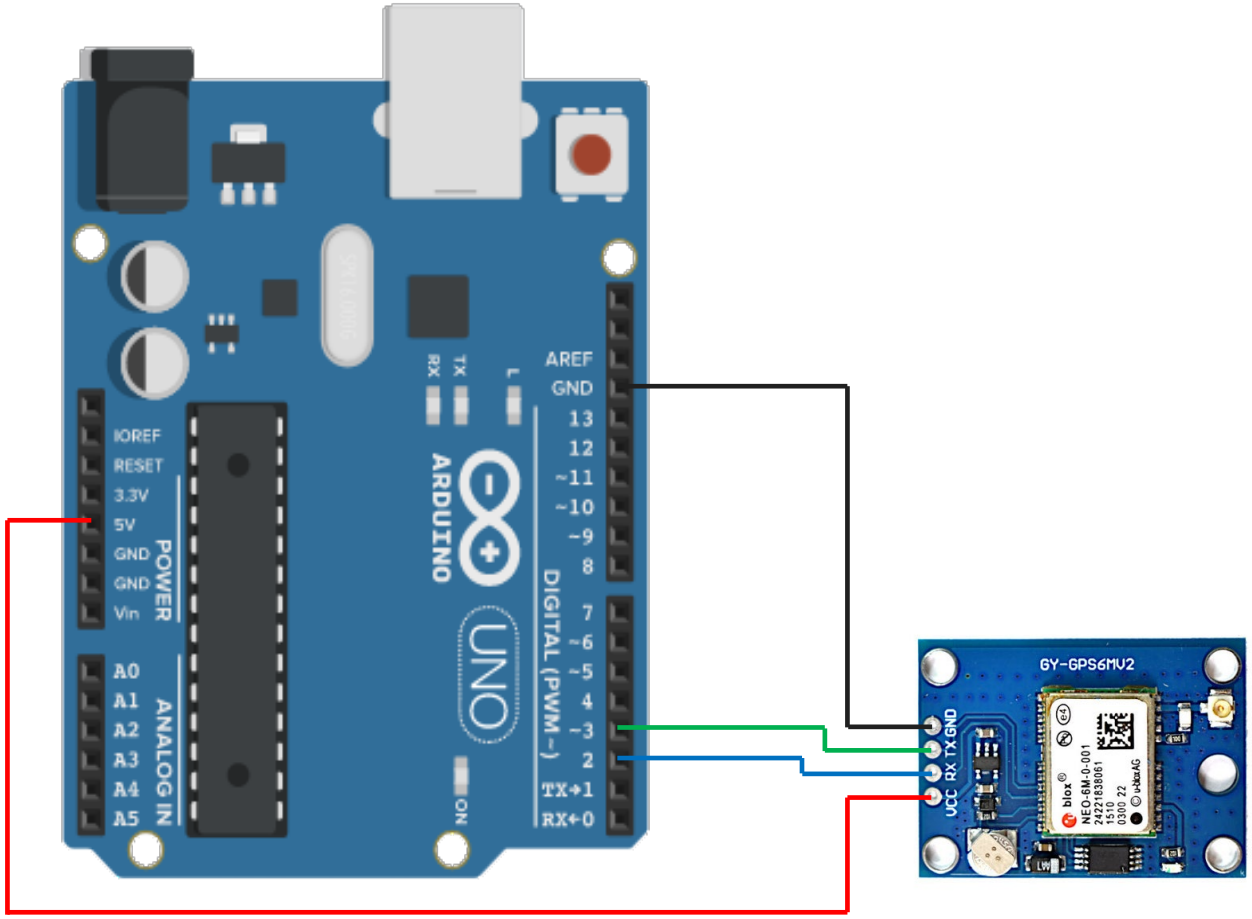
Latitude
Longitude

Returns latitude and longitude as output

Tracked Location

GPS Sensor – Interfacing with Arduino

- The interfacing of GPS Sensor with Arduino Uno is as shown in figure.
- Here, Vcc and Gnd pins of GPS Sensor are connected to 5V and Gnd of Arduino board.
- As the GPS communicates with Arduino serially, we need to connect Tx and Rx pin of GPS with serial pins of Arduino.
- But, we want the data received from the GPS sensor in serial monitor also to read the current location.
- Therefore, we need to use any digital pins of Arduino as serial transmit and receive pins which is done by software serial.



- Here, the Tx and Rx pins of GPS are connected with pins 3 and 2 respectively.

GPS Sensor – Code explanation

- The code in Arduino for GPS sensor can be written as following:

GPS_sensor.ino

```
1  #include <TinyGPS++.h>
2  #include <SoftwareSerial.h>
3  static const int RXPin = 3, TXPin = 2;
4  static const uint32_t GPSBaud = 9600;
5  TinyGPSPlus gps;
6
7  SoftwareSerial ss(RXPin, TXPin);
8
9  void setup()
10 {
11  Serial.begin(115200);
12  ss.begin(GPSBaud);
13 }
```

Includes the TinyGPS++ library to use built in functions for reading GPS sensor.

Includes the SoftwareSerial library to use software serial communication

Defining *gps* as object of TinyGPSPlus

Defining *ss* as object of SoftwareSerial

GPS Sensor – Code explanation (Cont.)

GPS_sensor.ino

```
14 void loop()
15 {
16  while (ss.available() > 0)
17    gps.encode(ss.read());
18    displayInfo();
19
20    if (millis()>5000 &&
21                    gps.charsProcessed()<10)
22    {
23       Serial.println("No GPS detected: Check
24  wiring.");
25       while(true);
26    }
27 }
```

The syntax reads the data serially from GPS and *gps.encode( )* function encodes that data as per the format.

GPS Sensor – Code explanation (Cont.)

GPS_sensor.ino

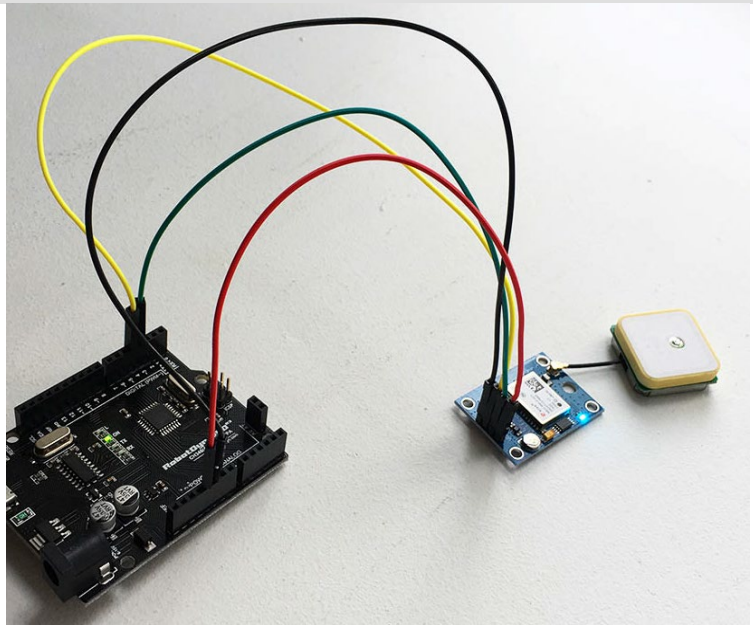
```
28 void displayInfo() {
29   Serial.print("Location: ");
30   if (gps.location.isValid())
31   {
32     Serial.print(gps.location.lat(), 6);
33     Serial.print(",");
34     Serial.print(gps.location.lng(), 6);
35   }
36   Serial.println();
37 }
```

gps.location.isValid( ) function returns "TRUE" if the location is in valid form

gps.location.lat( ) function returns the current latitude value and it is printed with 6 decimal point precision.

gps.location.lng( ) function returns the current longitude value and it is printed with 6 decimal point precision.

GPS Sensor – Output

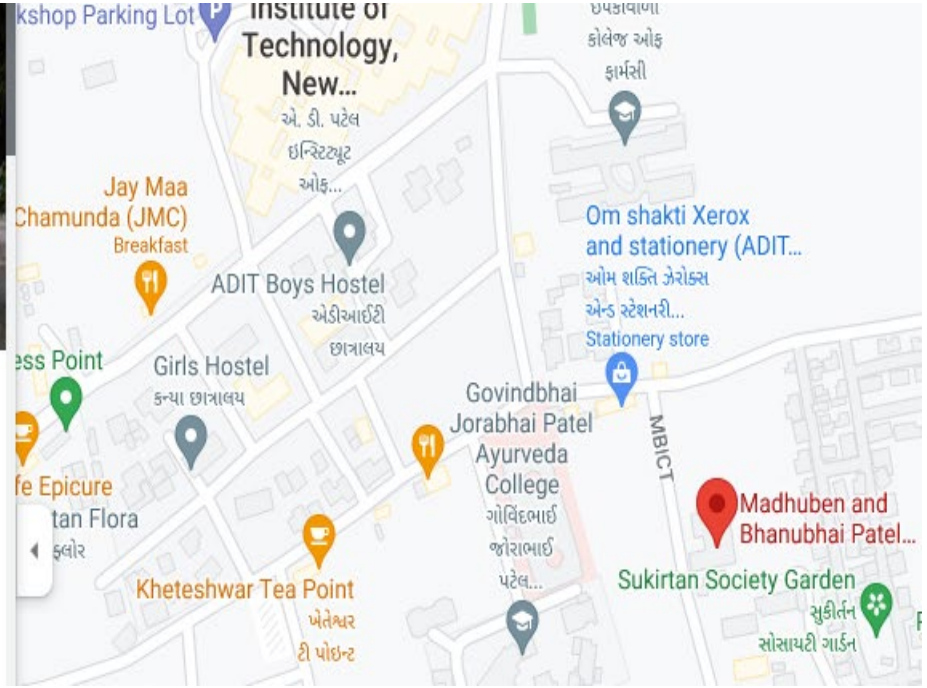


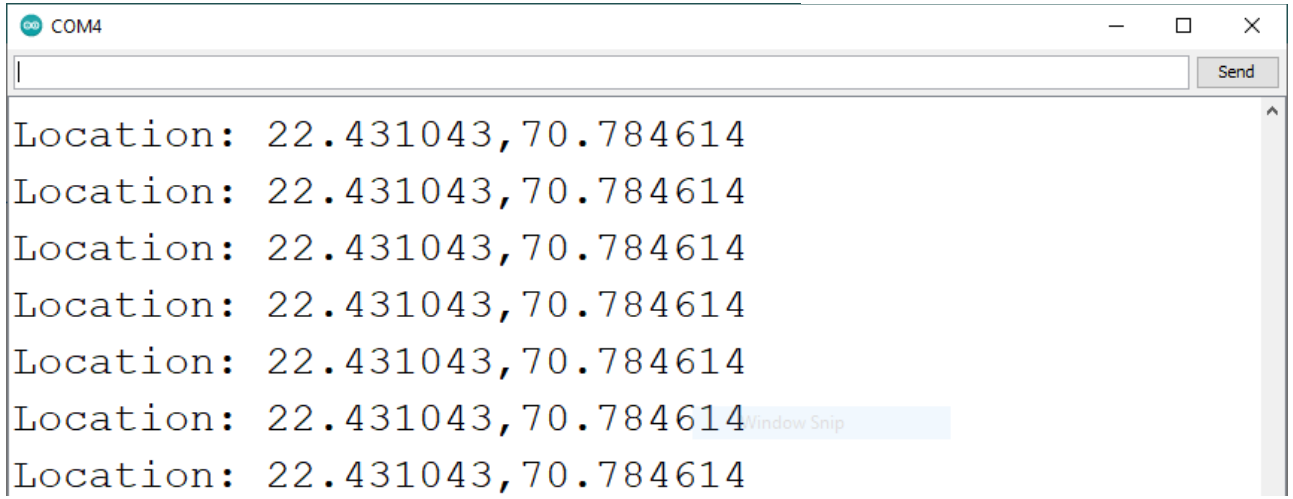
Madhuben and Bhanubhai Patel Institute of Technology - CVM University

મધુબેન એન્ડ ભાનુભાઈ પટેલ ઇન્સ્ટિટ્યૂટ ઓફ ટેક્નોલોજી - સીવીએમ યુનિવર્સિટી

4.4 ★★★★½ 209 reviews

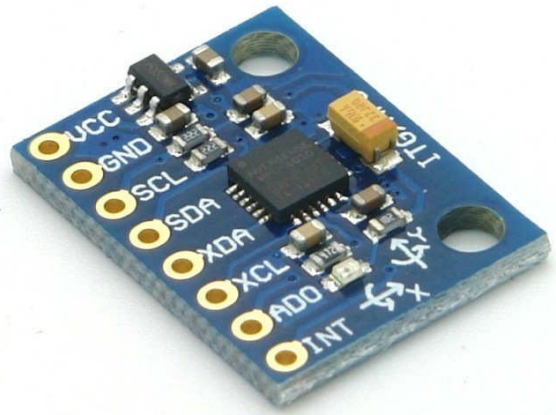
College





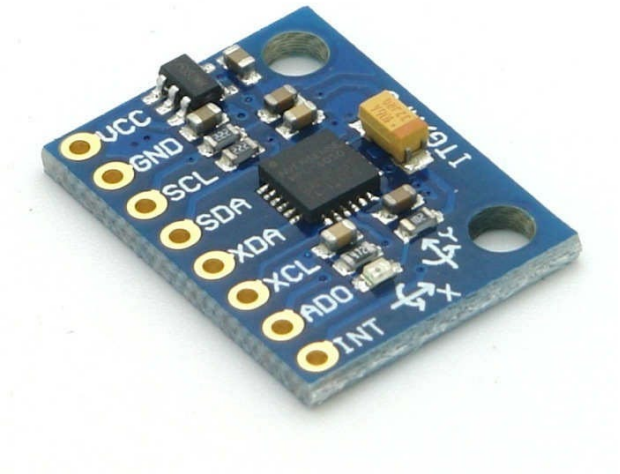
Gyro Sensor

- The Gyro sensor is very useful in **wearable devices.**
- It is used to find the position or angle of rotation of the body.
- It mainly senses
  - Rotational motion
  - Changes in orientation
- The Gyro sensor communicates with microcontroller with I2C (Inter-Integrated Communication).
- I2C is the most sophisticated two-wire communication protocol.
- It works with the device address and its respective data frames.
- Normally, I2C is used when more numbers of devices are connected.



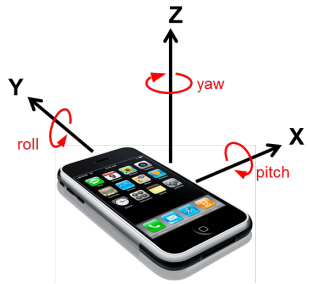
Gyro Sensor (Cont.)

- The pin out of Gyro sensor MPU6050 are as following.
  - Vcc – connected to 5V
  - GND – connected to ground
  - SCL – Serial Clock Line
  - SDA – Serial Data Line
  - XDA – Auxiliary Serial Data
  - XCL – Auxiliary Serial Clock
  - AD0 – This pin can be used to vary the address incase of multiple sensors used
  - INT – Interrupt pin available for certain application



MPU-6050 Gyro Sensor – working principle

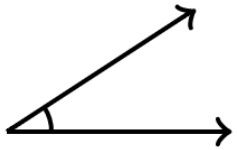
- The MPU 6050 senses the gravitational force applied to it.
- Place the sensor in device whose inclination is to be measured.
- Obtain the values of x, y and z coordinates.
- Determine the elevation angle by performing mathematical caluculations



Sensor placed for finding Inclination

x,y,z : 507 506 617
x,y,z : 507 504 616
x,y,z : 506 505 617
x,y,z : 506 506 618

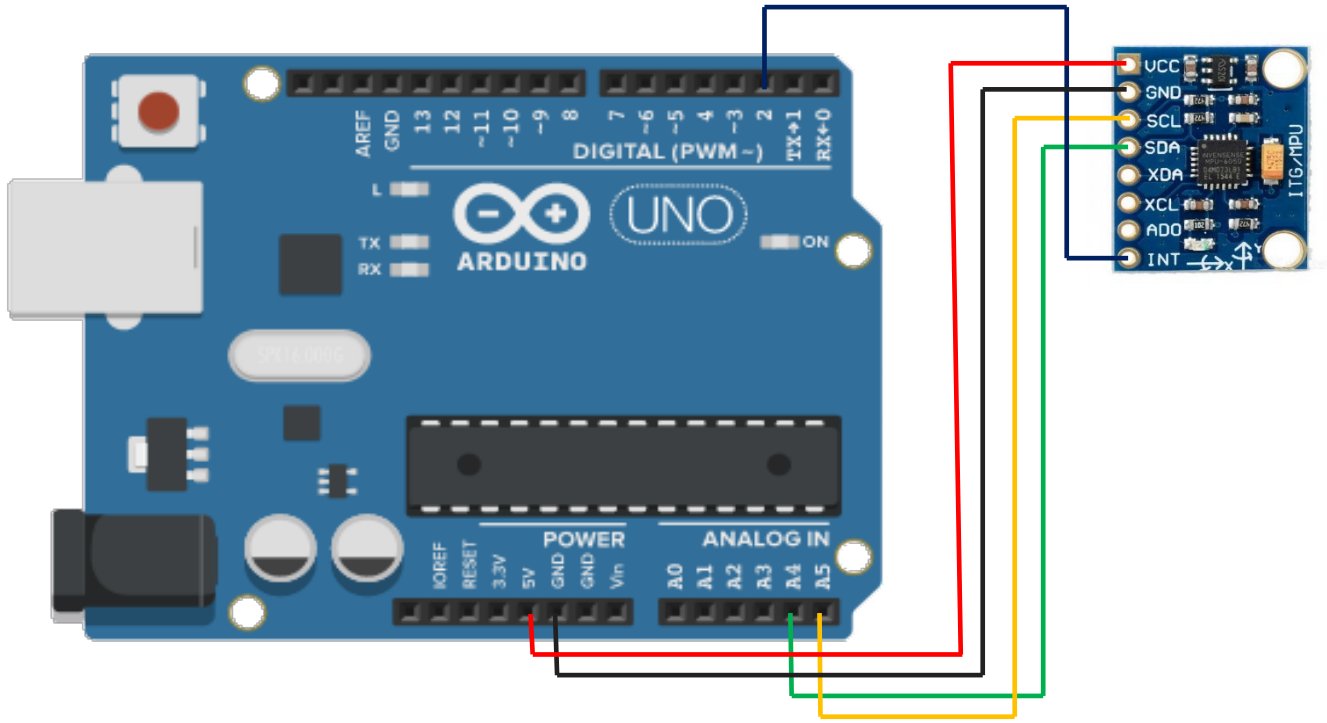
x, y, and z coordinates



Determination of Angle

MPU-6050 Gyro Sensor – Interfacing with Arduino

- The interfacing of MPU-6050 with Arduino Uno is as shown in figure.
- Here, Vcc and Gnd pins of MPU6050 is connected to 5V and Gnd of Arduino board.
- The SCL and SDA pins are used for I2C communication. These pins are available in Arduino with A5 and A4 pins as second functionality.
- Hence, SCL and SDA are connected to A5 and A4 pins of Arduino.



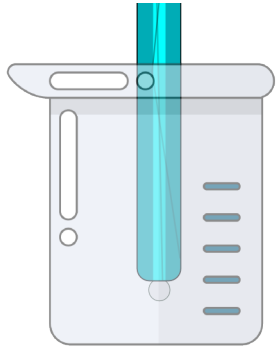
- The INT pin of MPU6050 gives interrupt when angular velocity increases than defined threshold value. So interrupt pin of MPU6050 is connected to external interrupt by Arduino supported at digital pin 2.

PH Sensor

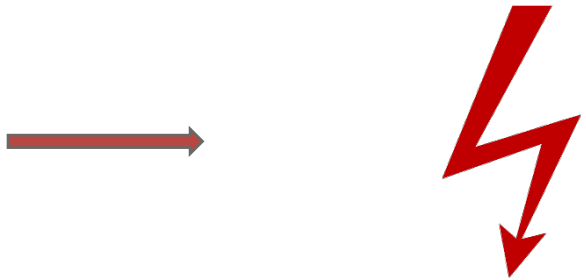
- The pH sensor is used to detect hydrogen ions concentration of a liquid.
- pH sensor are mostly used in laboratories to test the acidity of solution.
- By the use of pH sensor we can identify the pH of a solution and whether it is acid or base.

pH Sensor – Working principle

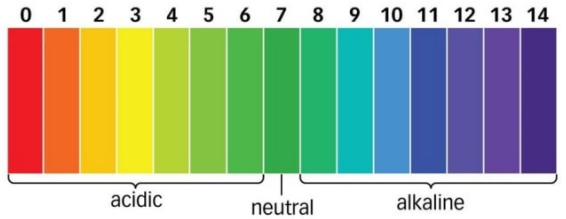
- When this sensor is placed in a solution, the smaller ions penetrate the boundary area of glass and the lager ions remain in the solution. This creates potential difference.
- The pH meter measures the difference in electrical potential between the pH electrodes.
- The potential difference generates different analog values for different liquids.
- By knowing the analog value of standard water, the pH value of other liquid can be determined.



pH Sensor with Electrodes



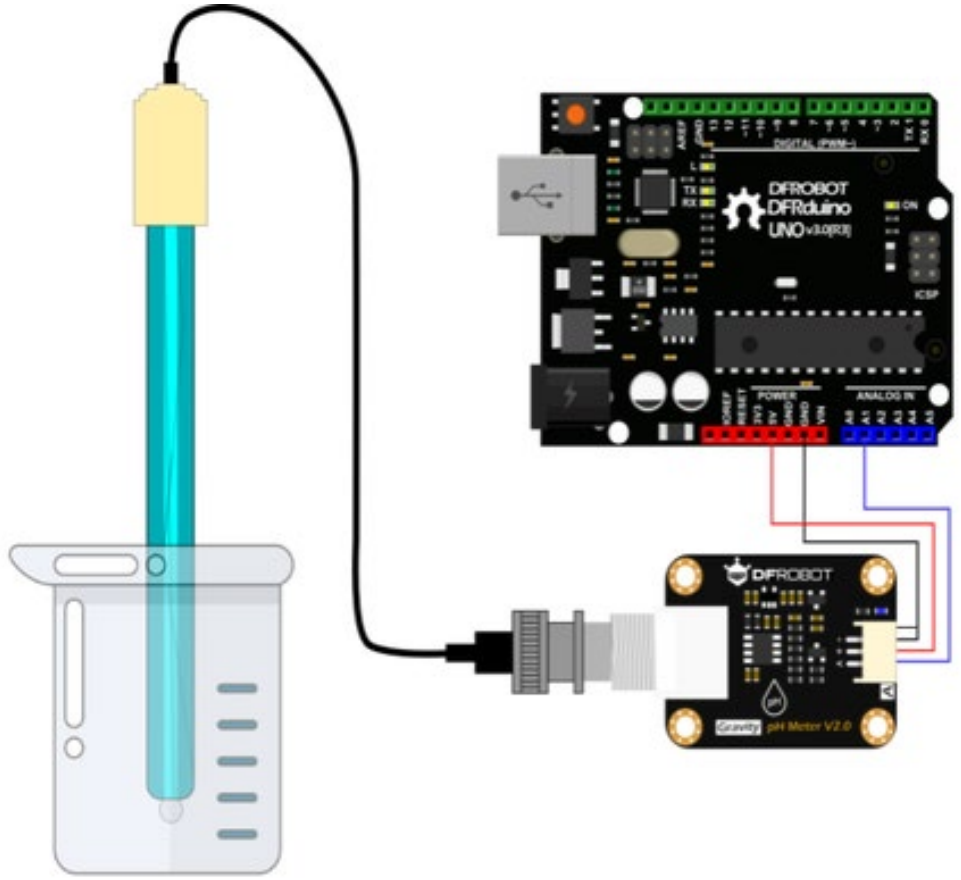
Output Voltage



Determination of alkalinity of the liquid

pH Sensor – Interfacing with Arduino

- The interfacing of pH sensor with Arduino is as shown in figure.
- Here, Vcc and Gnd pins of pH Sensor are connected to 5V and Gnd of Arduino board.
- The output pin of pH sensor is connected to analog pin of Arduino.
- The connection of pH sensor is done with voltage converter via BNC pin.



pH Sensor – Code explanation

- The code in Arduino for pH sensor can be written as following:

PH_sensor.ino

```
1  #define SensorPin A0
2  #define Offset 0
3  unsigned long int avgValue;
4  float b;
5  int buf[10],temp;
6
7  void setup()
8  {
9    pinMode(SensorPin,INPUT);
10   Serial.begin(9600);
11 }
```

Defining *offset* for correction of pH value. The offset value can be obtained by measuring the pH of standard water.

Defining *buf* as an integer array.

pH Sensor – Code explanation (Cont.)

PH_sensor.ino

```
12 void loop()
13 {
14   for(int i=0;i<10;i++) {
15     buf[i]=analogRead(SensorPin);
16     delay(10);
17   }
18   for(int i=0;i<9;i++) {
19     for(int j=i+1;j<10;j++) {
20       if(buf[i]>buf[j])
21       {
22         temp=buf[i];
23         buf[i]=buf[j];
24         buf[j]=temp;
25       }
26     }
27   }
```

Taking 10 different values of analog voltage from the pin where sensor is connected

Sorting the elements of array *buf* in ascending order.

pH Sensor – Code explanation (Cont.)

PH_sensor.ino

```
28    avgValue=0;
29    for(int i=2;i<8;i++)
30      avgValue+=buf[i];
31    avgValue = avgValue/6;
32    float phValue=(float)((avgValue)*
33                                  5.0/1024);
34    phValue=3.5*phValue+Offset;
35    Serial.print("pH: ");
36    Serial.print(phValue,2);
37    Serial.println(" ");
38    delay(800);
39  }
```

Avoiding lower and upper two (total four) values of array and taking the average of elements of rest of the array.

Converting the average value into equivalent millivolts.

Converting millivolts to its equivalent pH values as given in datasheet and adjusting with offset.

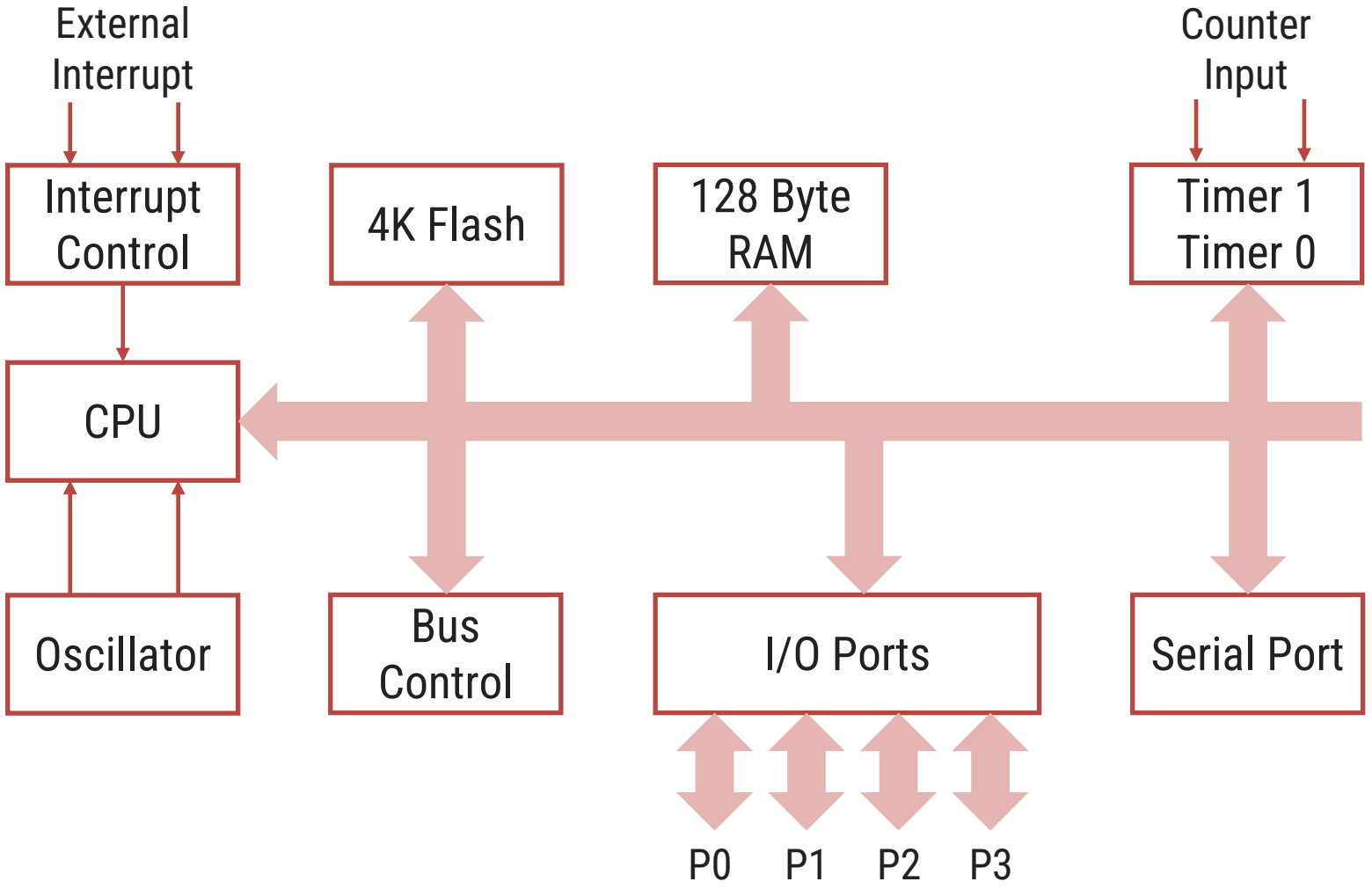
Other Microcontrollers

Section - 3

8051 Microcontroller Architecture

- 8051 was built by Intel but other companies are also permitted to make 8051 microcontrollers with same features and compatible set of instructions.

- The diagram shows Architectural representation of 8051 microcontroller.



Features of 8051 Microcontroller

1. 8- bit microcontroller.
2. Many general purpose and few special function registers (SFRs).
   - Two major 8 bit registers. A and B. Both are used in arithmetic operations mainly.
   - A is accumulator and it addressable.
   - There are 21 SFRs among them few are bit addressable. SFRs perform various dedicated operations.
   - Also, some control registers for timer, counter and interrupt fall in category of SFRs.
3. Four register banks having 8 registers in each bank.
4. Data pointer register which is 16 bit made from combinations of two 8 bit registers – DPH and DPL.
5. 16 bit Program Counter.
6. 8 bit Program Status Word(PSW) and used as Flag register.
7. Internal ROM and EPROM.
8. Internal user accessible RAM of 128 Bytes

Features of 8051 Microcontroller

9. There are four ports P0, P1, P2 and P3 that can be configured as input and output.
10. Two 16 bit timer/counter – T0 and T1.
11. Full duplex serial communication and dedicated serial buffer register SBUF.
12. Supports interrupt programming.
13. Oscillator and Clock circuits are built in.
14. Easier and simpler instruction set.

Memory organization in 8051

- There are two categories of memory – program memory and data memory.

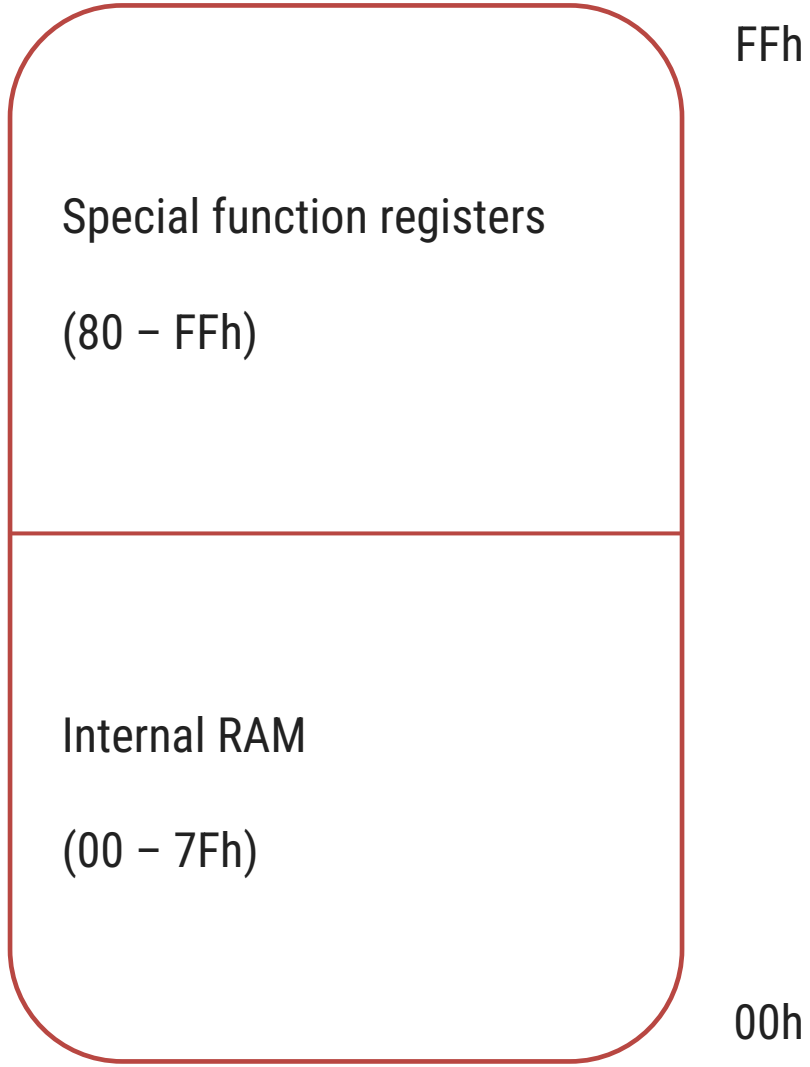
1. Program memory

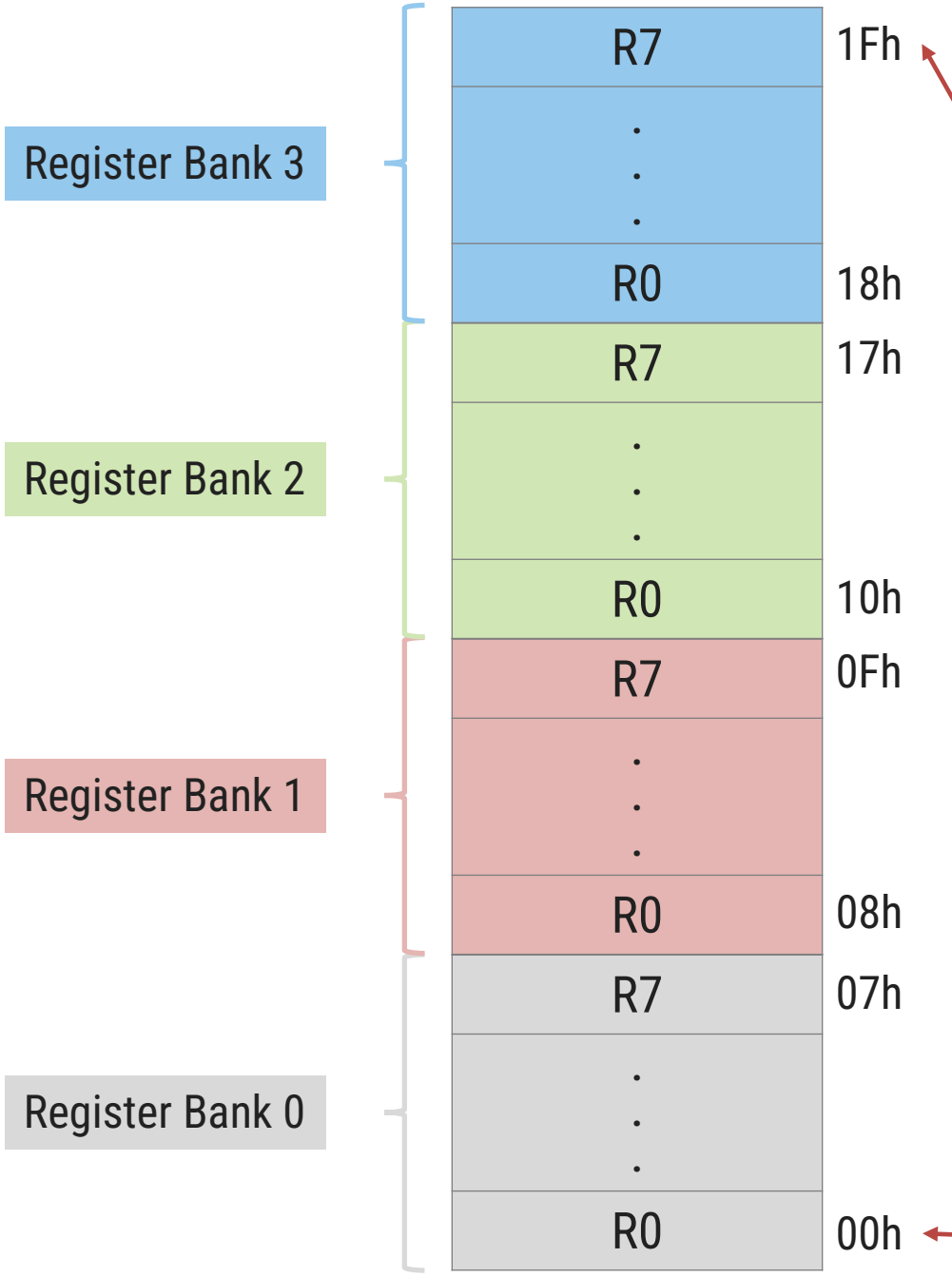
- In 8051, 4K bytes of program memory is available.
- It is normally referred as ROM and non volatile in nature.
- It is used to store following:
  1. Boot up programs
  2. Interrupt Service Routines (ISR)
  3. Macro Functions

2. Data memory

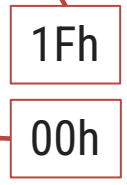
- Data memory is used to store temporary data.
- 8051 has 256 byte data memory neatly organized for various operations.
- The 256 byte data memory is divided into two parts

1. First 128 Bytes – Internal user accessible RAM
2. Last 128 Bytes – Special Function Registers (SFRs).

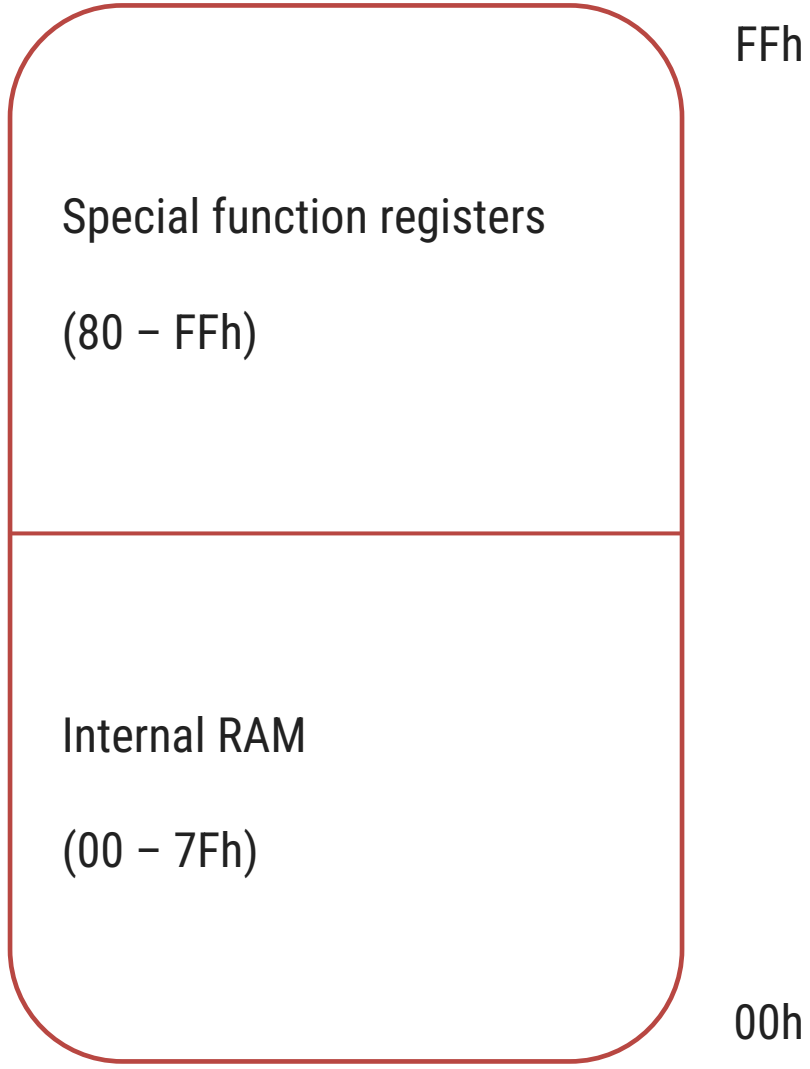


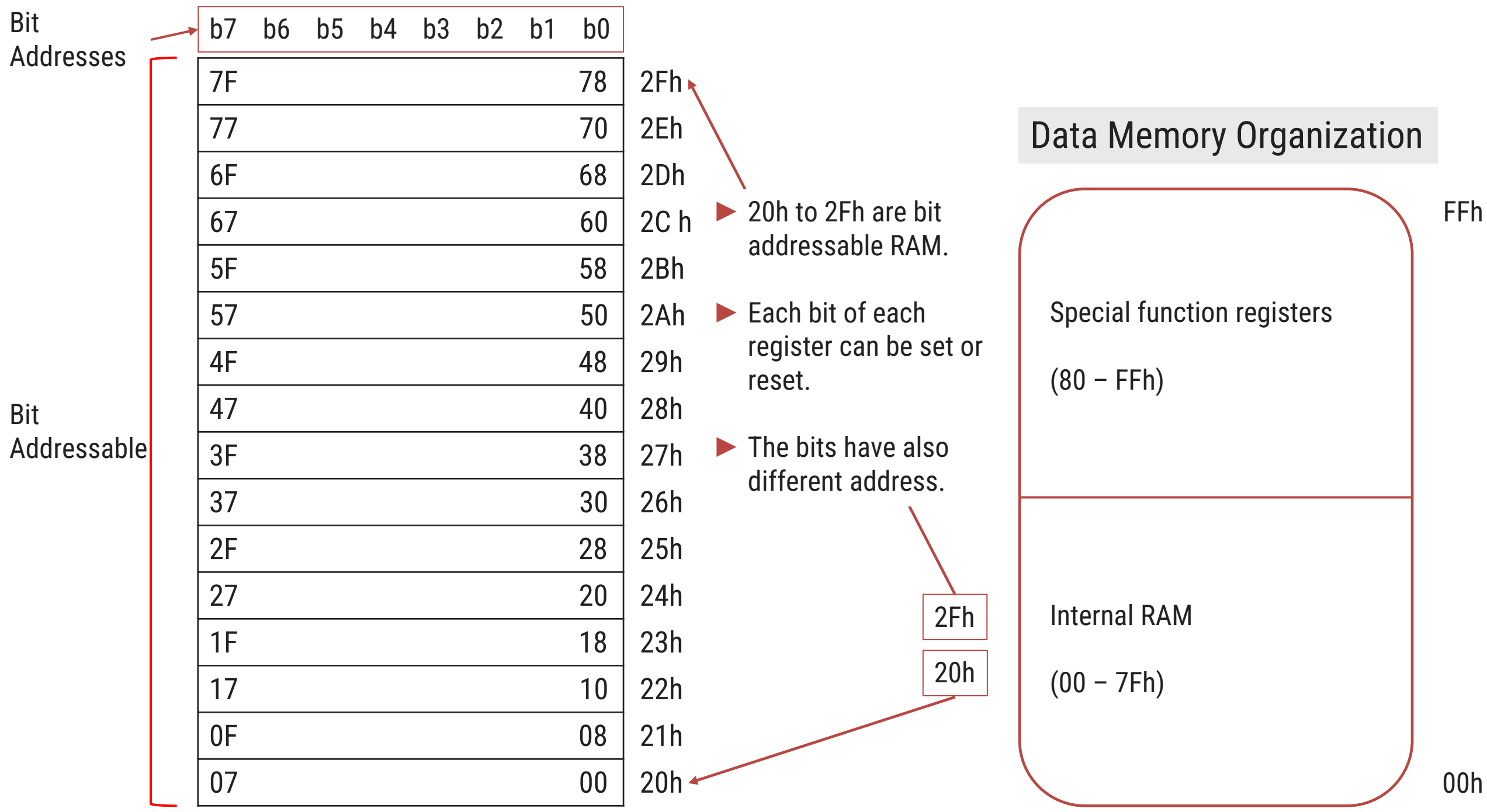


- The first 32 Bytes 00h to 1Fh are allocated for register banks.
- There are four register banks – register bank 0,1,2 and 3.
- Each register bank has 8 registers R0 to R7.

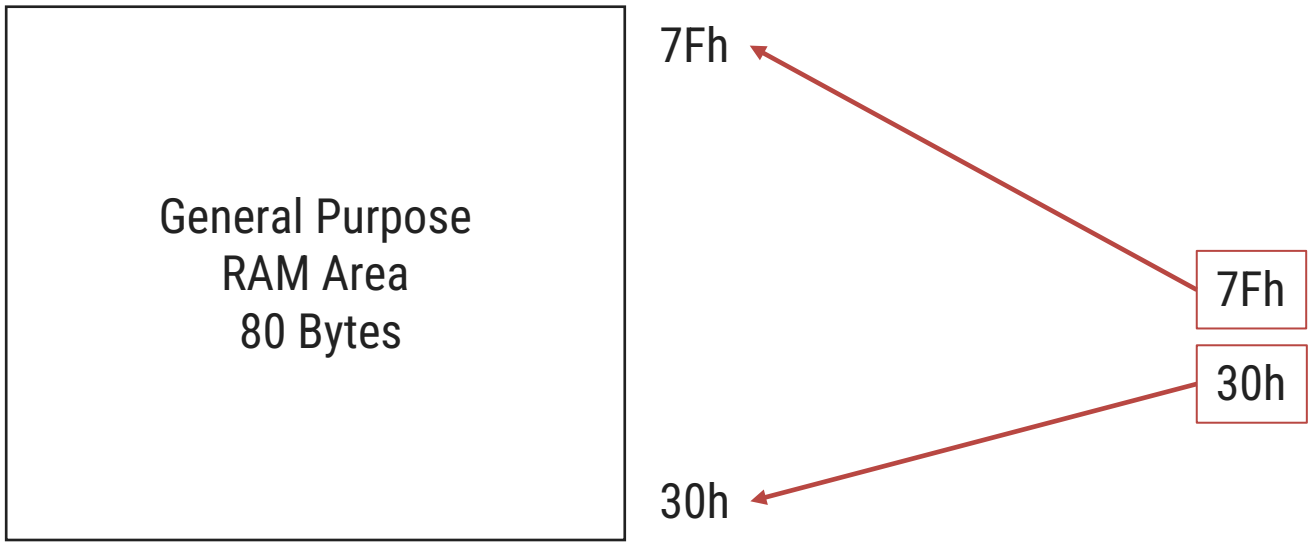


Data Memory Organization

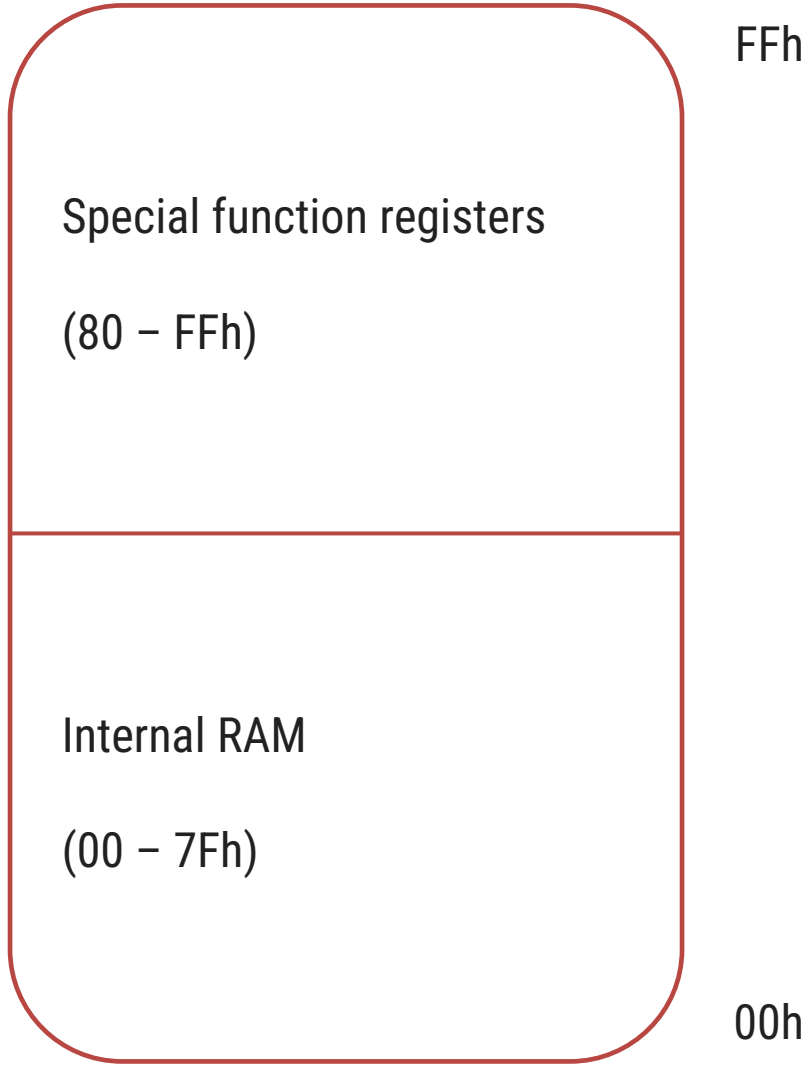



Bit Addresses
b7 b6 b5 b4 b3 b2 b1 b0
7F 78 2Fh
77 70 2Eh
6F 68 2Dh
67 60 2C h
5F 58 2Bh
57 50 2Ah
4F 48 29h
47 40 28h
3F 38 27h
37 30 26h
2F 28 25h
27 20 24h
1F 18 23h
17 10 22h
0F 08 21h
07 00 20h
Bit Addressable
20h to 2Fh are bit addressable RAM.
Each bit of each register can be set or reset.
The bits have also different address.
2Fh
20h
Data Memory Organization
FFh
Special function registers
(80 – FFh)
Internal RAM
(00 – 7Fh)
00h

- The last 80 bytes of Internal RAM that is 30h to 7Fh are used as General purpose RAM Area.
- Any general purpose data is stored in this area.



Data Memory Organization



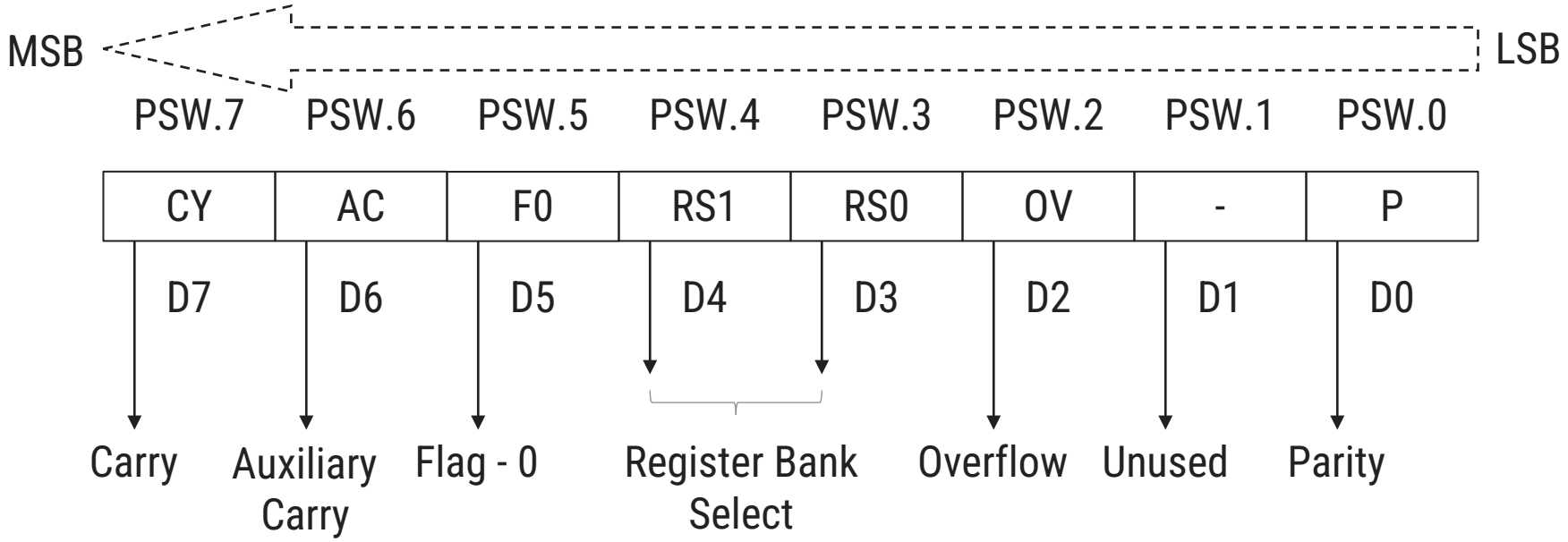
Special Function Registers

- There are 21 SFRs in 8051microcontroller.
- They are important set of registers used for dedicated operations in 8051 microcontroller.
- Some of the SFRs are bit addressable while others are not.
- The following list explains the register and their function. The bracket is representing its address in RAM.

1. Accumulator – A (E0h)
   - It is 8 bit register used in each Arithmetic and Logical operation.
   - It is bit addressable.
   - All the results of arithmetic or logical operations are stored in Accumulator
   - One operand in all operations such as addition, subtraction, division and multiplication will be accumulator.
2. Register B (F0h)
   - It is 8 bit register used in multiplication and division mainly.
   - It is bit addressable.

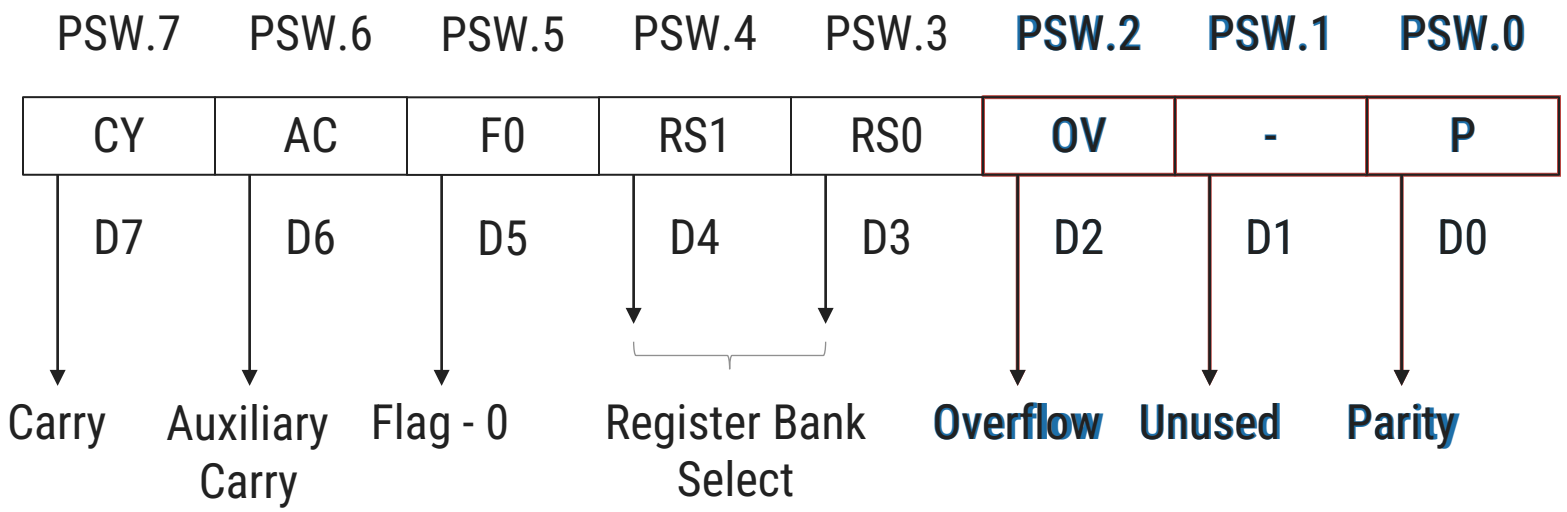
Special Function Registers (Cont.)

3. PSW Register (D0h)
   - PSW (Program Status Word) is consisting various flags of 8051.
   - These flags indicate whether the microcontroller works as expected or something is going wrong.
   - The format of PSW register is as shown in figure.
   - It is 8 bit register.
   - Also PSW is bit addressable and its bits are represented by PSW.X, where X represents bit number.
   - Bit 0 is LSB and bit 7 is MSB.



Special Function Registers (Cont.)

3. PSW Register (D0h)
   - P – Parity Flag (PSW.0)
     - It is D0 bit of PSW register. The parity flag is set or reset based on the result stored in accumulator. For odd number 1s in result, the flag is set otherwise it is reset.
   - PSW.1
     - It is not used and reserved for future usage
   - OV – Overflow Flag (PSW.2)
     - When the result of signed arithmetic operation is very large OV will be set.



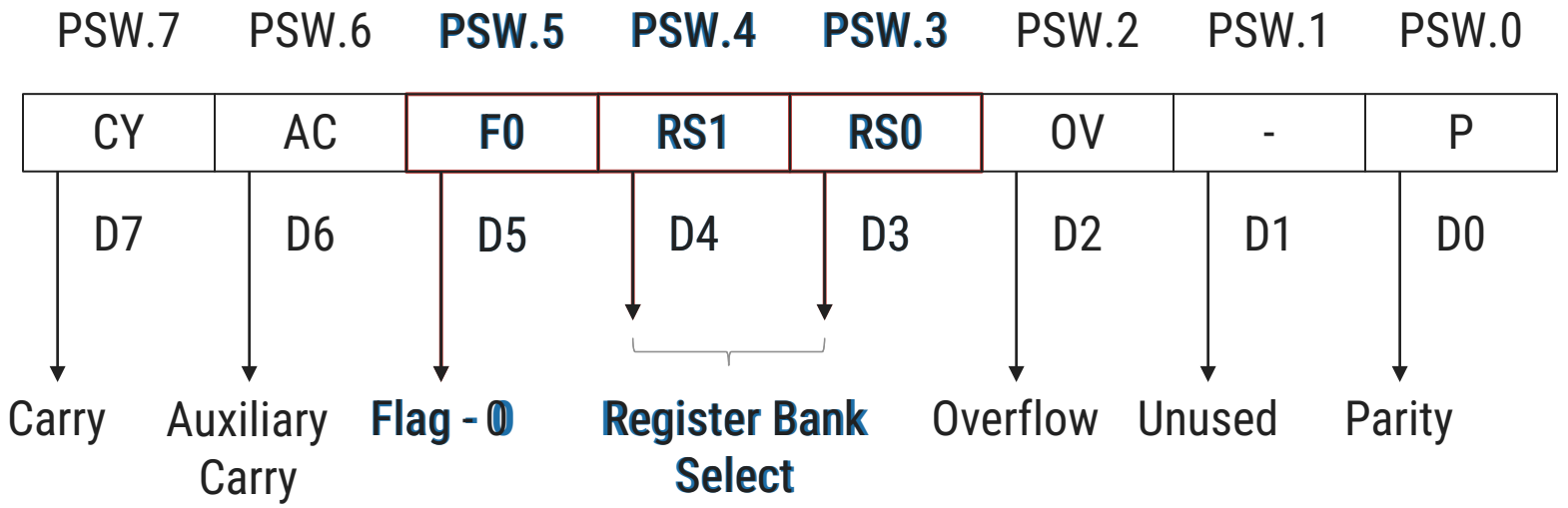
Special Function Registers (Cont.)

3. PSW Register (D0h)

- RS0 and RS1 – Register Bank Select (PSW.3 and PSW.4)
  - The four register bank available in Internal RAM area can be selected by RS0 and RS1 bits. The table shows combination of RS1 and RS0 to select a particular register bank.

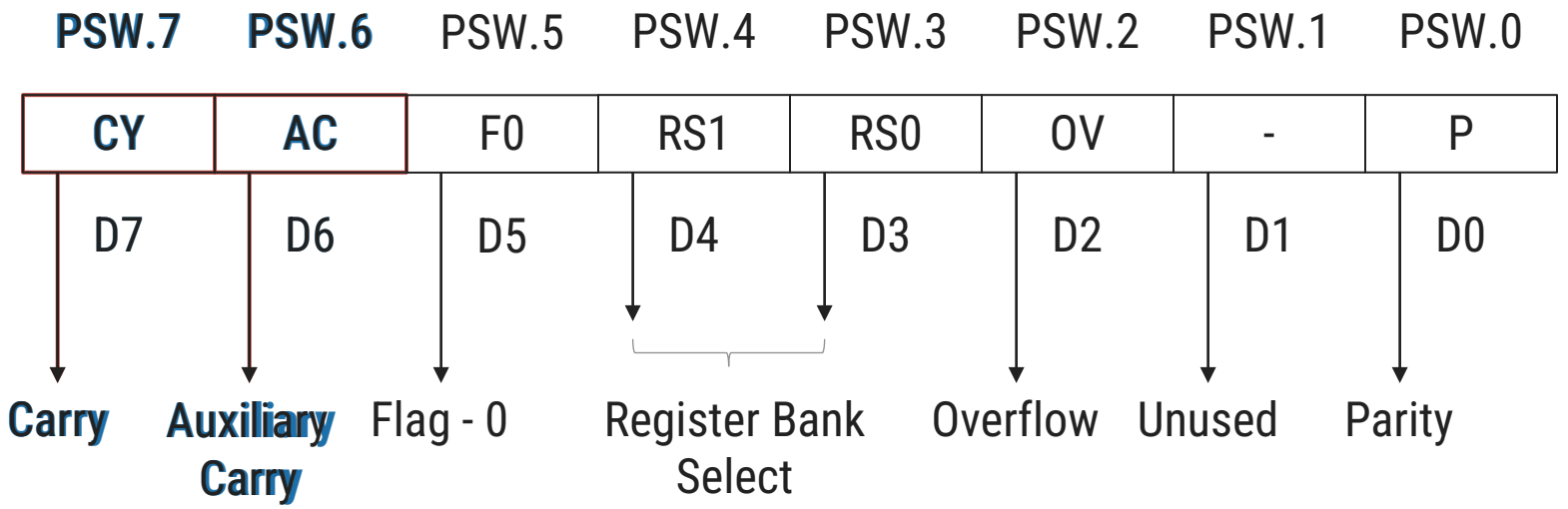
RS1	RS0	Bank Selected
0	0	Register Bank 0
0	1	Register Bank 1
1	0	Register Bank 2
1	1	Register Bank 3

- F0 – Flag 0 (PSW.5)
  - It is a general purpose bit that can be used by a user as per requirement.



Special Function Registers (Cont.)

3. PSW Register (D0h)
   - AC – Auxiliary Carry (PSW.6)
     - It is used in BCD operations. When there is a carry from D3 to D4 bit this flag is set otherwise it remains reset.
   - CY – Carry (PSW.7)
     - Carry flag is set to one when the arithmetic operation has carry generated from D7 bit. It is also used in shift operations.



Special Function Registers

- Timer and Counter registers related to timer and counter programming

4. TCON (88h) (Timer Control Register)
5. TMOD (89h) (Timer Mode Register)
6. TL0 (8Ah) (Timer0 Lower Byte)
7. TL1 (8Bh) (Timer1 Lower Byte)
8. TH0 (8Ch) (Timer0 Higher Byte)
9. TH1 (8Dh) (Timer1 Higher Byte)

- PORT register – These registers are used for PORT programming.

10. P0(80h) 11. P1(90h) 12. P2(A0h) 13. P3 (B0h)

- Interrupt programming registers

14. IE (A8h) (Interrupt Enable)
15. IP (B8h) (Interrupt Priority)

Special Function Registers

- Power control register

16. PCON (87h) (Power Control)

- Serial Communication Registers

17. SCON (98h) (Serial Control)
18. SBUF (99h) (Serial Buffer)

- Data Pointer Registers (DPTR) – This register is combination of two 8 bit registers DPH and DPL. It is used when data is to be stored in memory pointing out a particular location.

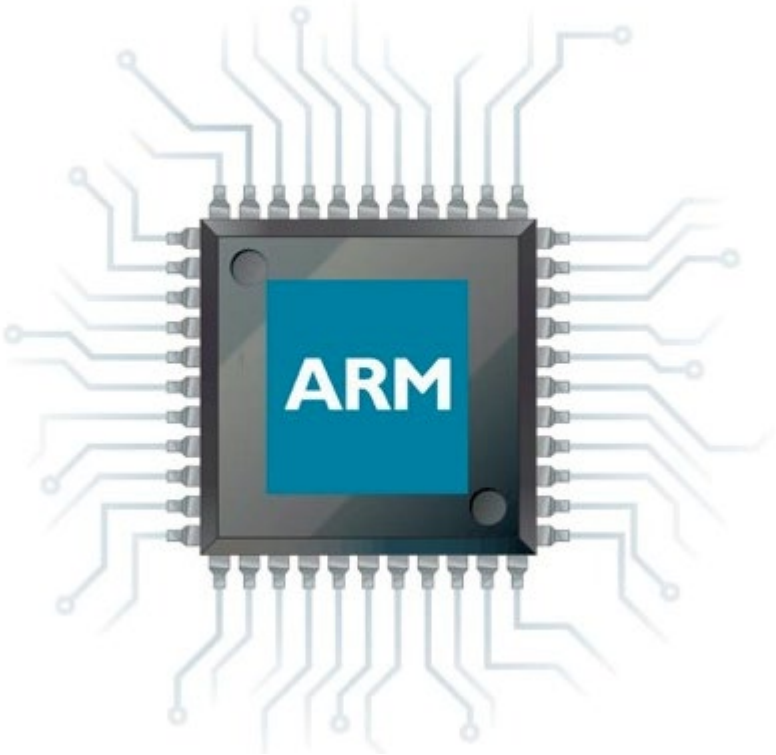
19. DPH (83h) (Data Pointer Higher byte)
20. DPL (82h) (Data Pointer Lower byte)

- Stack Pointer – It is used while storing the data into stack memory.

21. SP (81h) (Stack Pointer)

Advanced RISC Machine (ARM)

- An Advanced RISC Machine (ARM) is a processor which is family of CPUs based on Reduced Instruction Set Computer.
- The ARM processor is 32 bit processor developed for high-end applications that involve more complex computations and calculations.
- The ARM is based on RISC but has been enhanced for usage in embedded applications.

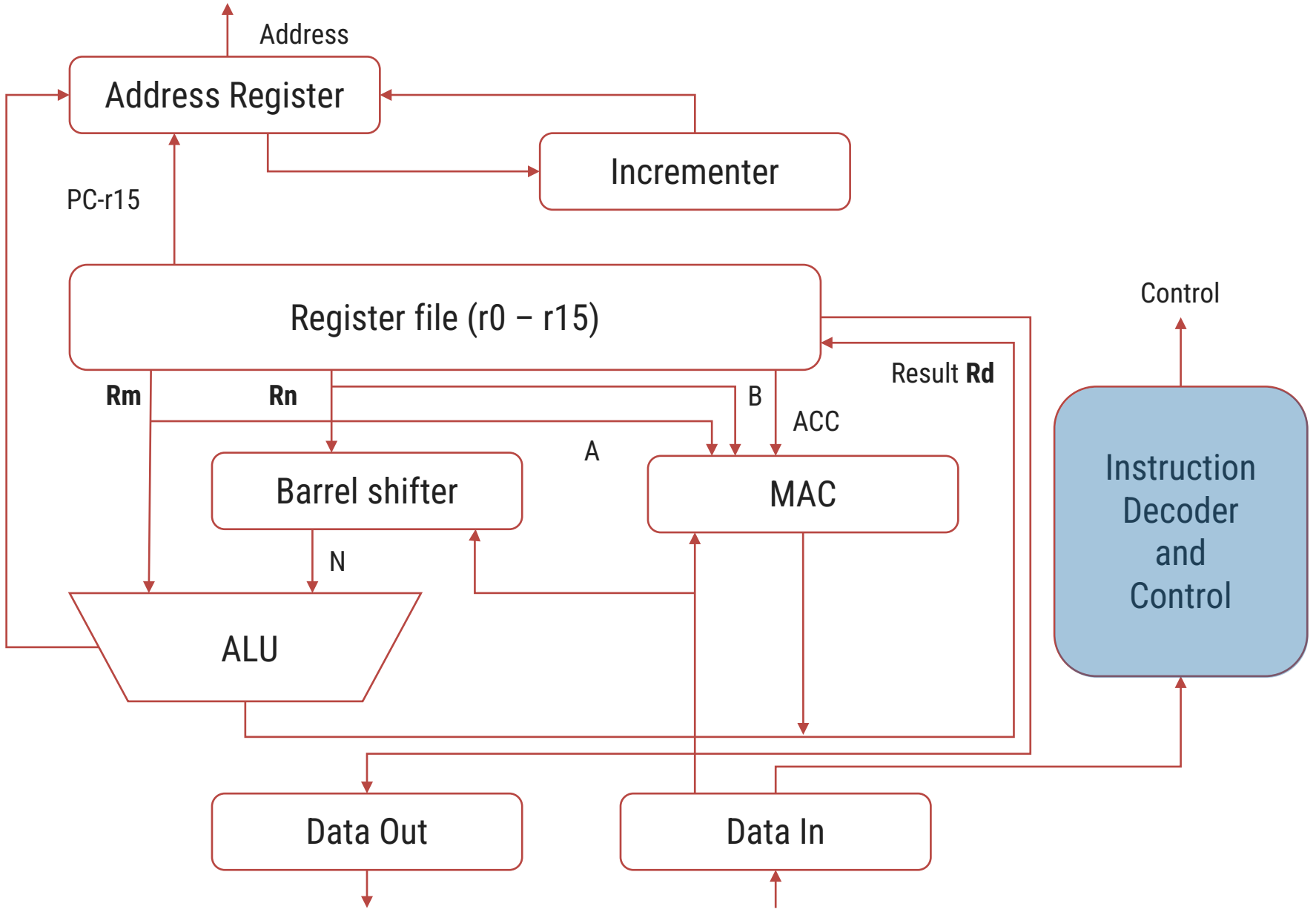


Features of ARM

- It has large uniform register files with load and store architecture. The load-store architecture allows a single instruction to perform load operation from main memory into register and store operation from a register into main memory.
- ARM is 32 – bit processor with reverse compatibility. So it has 16-bit and 8-bit variants embedded into 32 – bit processor.
- It has very good speed v/s power consumption ratio and high code density.
- It has a barrel shifter in the data path, which can maximize hardware usage available on the chip.
- Built in auto-increment and auto-decrement addressing modes.
- It has conditional execution instructions.

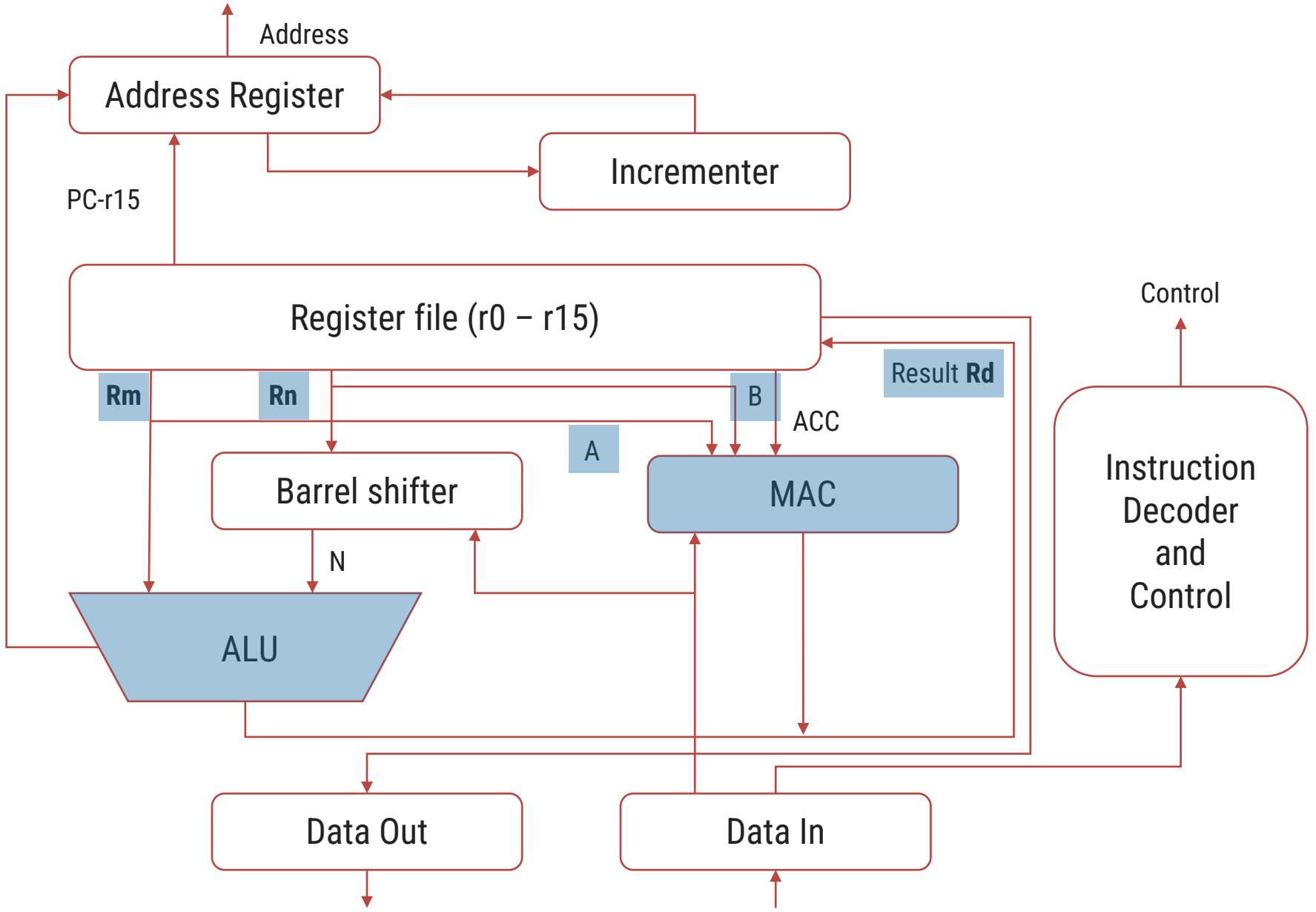
ARM Core Data Flow Model

- Figure shows ARM core flow diagram where the functional units are connected by data buses.
- The important blocks can be explained as following.
- Instruction Decoder
  - ➥ It decodes the instruction before execution.
  - ➥ There are three kinds of instruction set supported
  1. ARM Instruction set
  2. Jazelle Instruction set
  3. Thumb Instruction set



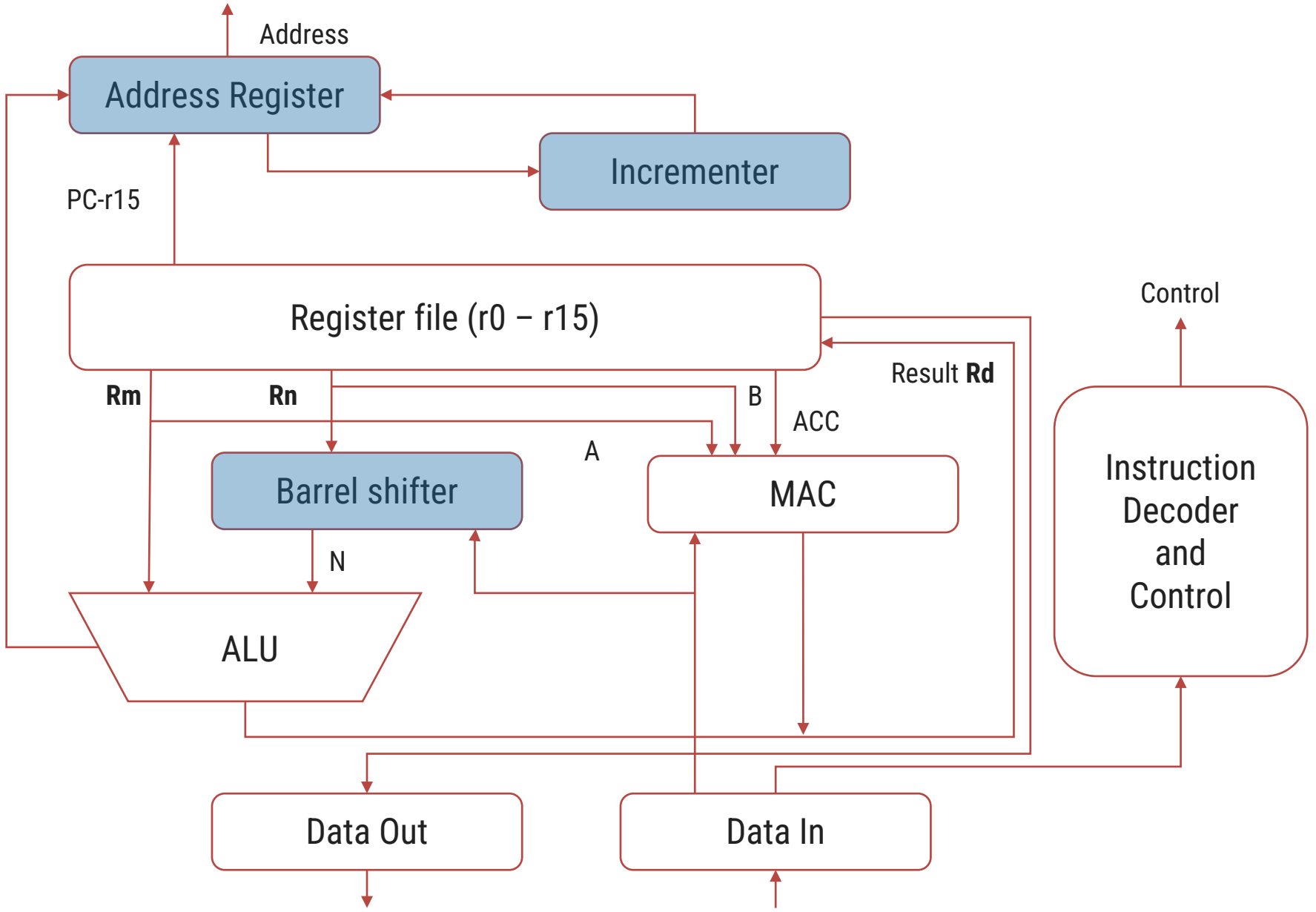
ARM Core Data Flow Model

- Rm, Rn and Rd
  - ARM has two source registers Rm and Rn and one destination register Rd.
- A and B buses
  - A and B buses are used to fetch the data from source register and provides for further operation.
- ALU and MAC
  - The computation is carried out in ALU or MAC (Multiply Accumulate) unit.



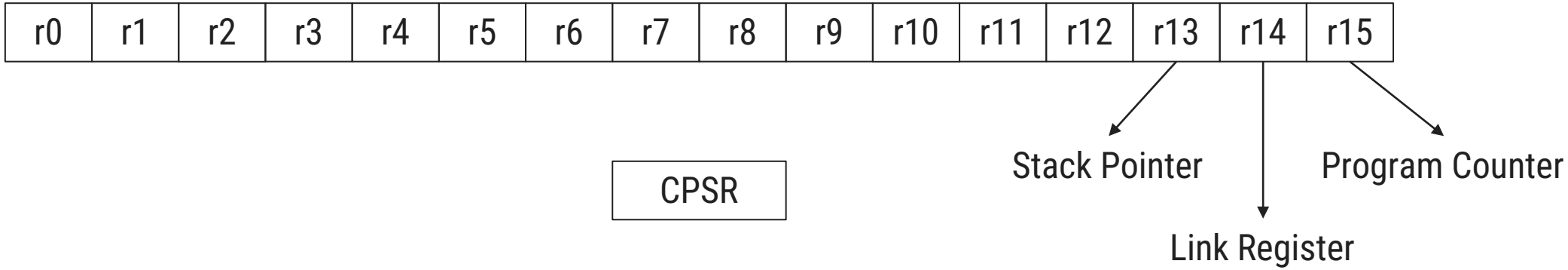
ARM Core Data Flow Model

- Barrel Shifter
  - Barrel shifter is often used for shift and rotate operation in a single clock cycle.
- Address register
  - The address register is used to hold the register and the address bus will carry the current address.
- Incrementer
  - The incrementer is used for auto-increment of address and stored back in the address register.



ARM Register Organization

- ARM has 13 general purpose registers (r0 to r12)
- All the registers are 32 bit in size.
- There are 3 SFRs in ARM
  - r13 (Stack Pointer)
  - r14 (Link Register)
  - r15 (Program Counter)
- There is one more important register in ARM – CPSR (Current Program Status Register).

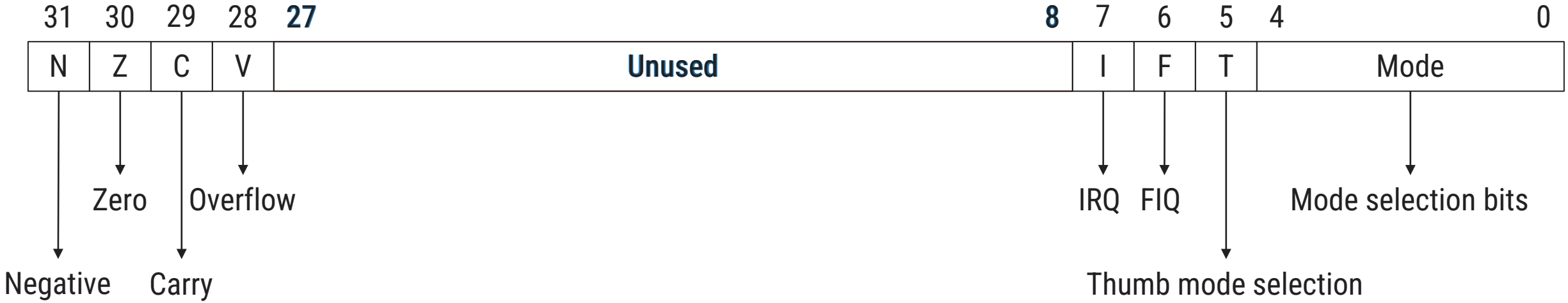


ARM Register Organization - SFRs

1. Stack Pointer (SP/r13)
   - The stack pointer holds the address of the top of the stack.
   - It is used for accessing the stack memory.
2. Link Register (LR/r14)
   - This register stores the address of next instruction to be executed when the interrupt is generated and the processor will jump to execute the service routine or subroutine.
   - After executing the subroutine, this register will return the address from where the program was interrupted.
3. Program Counter (PC/r15)
   - The program counter is used to sequence the flow of execution of program.
   - It holds the address of the next instruction to be executed.

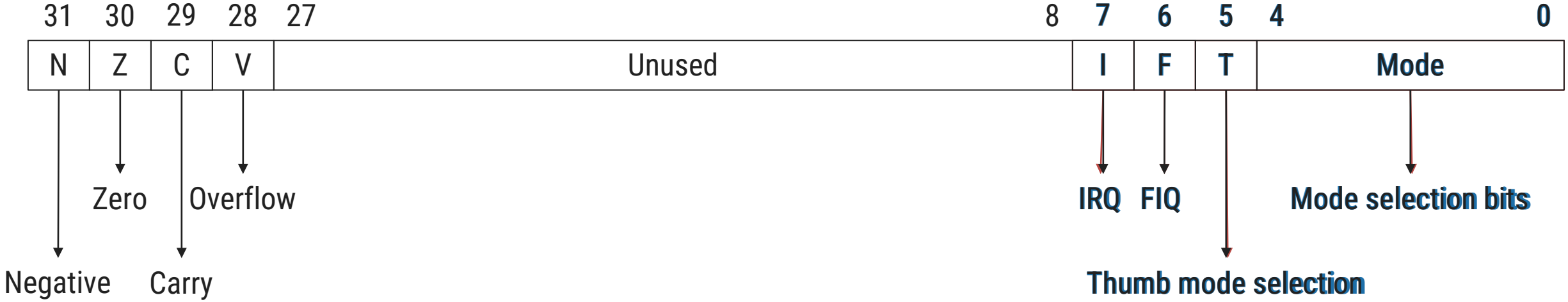
Current Program Status Register (CPSR)

- CPSR is the status register or flag register equivalent to PSW in 8051 but CPSR is of 32 bits.
- The format of CPSR register is shown in figure.
- Among 32 bits, 20 bits are unused or reserved for future expansion.



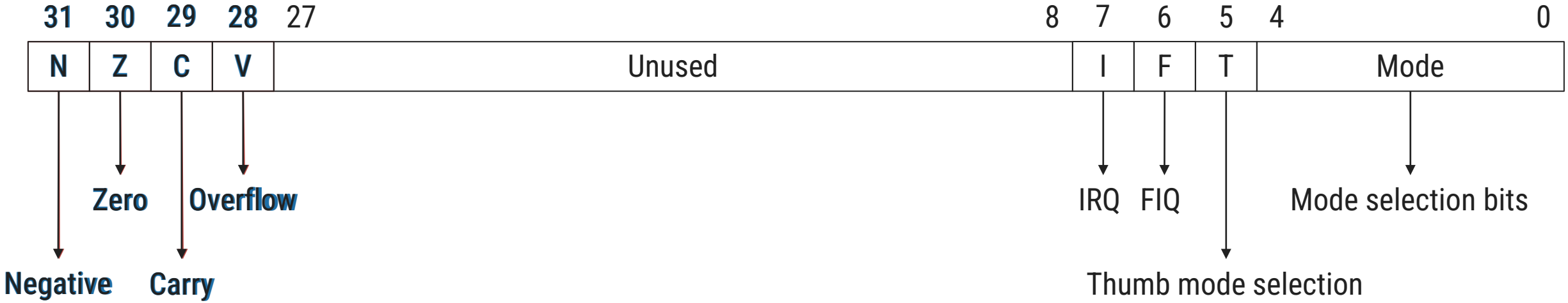
Current Program Status Register (CPSR)

- Mode selection bits
  - ARM can operate in seven different modes. Total 5 bit 0 to 4 are assigned for mode selection.
- T – Thumb mode selection
  - If this bit is set to 1, ARM works in Thumb mode else ARM works in normal mode.
- F – FIQ Interrupt Mask
  - This bit is set to zero to enable FIQ interrupts.
- I – IRQ Interrupt Mask
  - This bit is set to zero to enable IRQ or normal interrupts.



Current Program Status Register (CPSR)

- V – Overflow Flag
  - When result of a signed operation results in an overflow, this flag is set to 1.
- C – Carry Flag
  - This flag is set to 1 if the result of an ALU operation generates carry.
- Z – Zero Flag
  - This flag is set to 1 if the result of an ALU operation is zero.
- N – Negative Flag
  - This flag is set to 1 if the result of an ALU operation is negative.



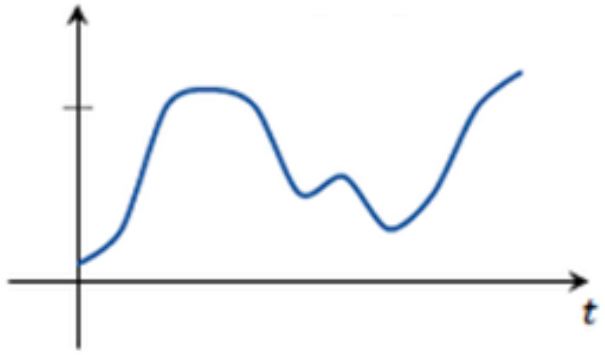
Difference between Microprocessor and Microcontroller

Difference between Microprocessor and Microcontroller

	Microprocessor	Microcontroller
Application	It is used in applications of General purpose.	It is used in specific applications.
Peripherals	Peripheral devices (like memory, I/O ports, ADC/DAC etc.) are required for the operation of Microprocessor based system.	Peripheral devices are not required to operate.
Size of memory	Microprocessor based system will have large size of RAM and program memory (MB/GB).	Microcontroller based system will have smaller size of RAM and program memory (KB/MB).
Cost	The cost of the microprocessor is high compared to the microcontroller.	It is cheaper than Microprocessor.
Complexity	The structure of Microprocessor based system is complex compared to Microcontroller	The structure of Microcontroller based system is simpler.
Power Consumption	The power consumption for the microprocessor is high.	The power consumption for the microcontroller is less.

Difference between Analog and Digital

Difference between Analog and Digital Signal

Analog Signals	Digital Signals
Analog signals can have any possible values in the given range.	Digital signals can have only discrete values in the given range.
Examples: All physical quantities (like temperature, light intensity) measured by different analog sensors, sound waves	Examples: Count of persons entered in premises, roll numbers etc.
	

Analog vs Digital Sensors

Factors	Analog Sensors	Digital Sensors
Data Transmission	Deterioration by noise	Noise immune without deterioration
Signal	Continuous Signal is representing physical measurements	Digital signal representing discrete-time signals generated by digital modulation
Bandwidth	Lower Bandwidth	Higher Bandwidth
Power	Takes large power	Negligible Power
Waves	Represented by Sine Waves	Denoted by Square Waves
Impedance	Impedance is Low	High Impedance of order 100 megaohm
Errors	Observational error occurs	Free from Observational error

Comparison of Flash, SRAM and EEPROM

- Flash memory is also known as program memory where the program for any microcontroller or IC is stored.
- SRAM (Static Random Access Memory) is where the program creates and manipulates variables and stores temporary data.
- EEPROM(Electronically Erasable Programmable Read Only Memory) is the space where programmer need to store long term information.

Microcontroller	Microprocessor
A microcontroller is a specialized form of a microprocessor	The microprocessor is designed to be general-purpose.
It is cost-effective.	It is a silicon chip
It is self-sufficient.	It is a dependent unit
The microcontroller is used to perform a particular tasks.	The Microprocessor is used to perform a certain task.
Its power consumption is low.	Its power consumption is high.
It contains CPU, RAM, ROM, Registers, Timer and input/output ports.	It requires a combination of timers, controllers memory chips.
Its size is smaller.	Its size is larger.
It is a chip which is called single chip computer.	It is a general purpose device which is called a CPU.
Microcontroller have no advantage of designing RAM, ROM, I/O port.	It have advantages of versatility such that designer can decide the aount of RAM, ROM, I/O port as needed.
Its microprocessors processing power is lower than microprocessor.	Its processing power is higher.
It uses Harvard Architecture.	It uses Von Neumann Architecture.
It's system cost is low.	It's system cost is high.
Each instruction needs an internal operation.	Each instruction needs an external operation.
For Example- Television.	For Example- Personal Computers.

Feature	Microcontroller (µC)	Microprocessor (µP)
Purpose	Designed for specific embedded system applications	Designed for general-purpose computing applications
Architecture	Single-chip computer system with on-board memory, peripherals, and I/O interfaces	CPU with minimal on-board memory, peripherals, and I/O interfaces
Integration level	Highly integrated	Less integrated
System architecture	Single-chip system	CPU + support chips
Processing power	Lower power	Higher power
Instruction set	Fixed instruction set	More flexible
On-board memory	On-chip memory	No on-board memory
Input/Output (I/O)	More I/O ports	Fewer I/O ports
Peripheral devices	On-board peripherals	External peripherals
Cost	Lower cost	Higher cost
Power consumption	Lower power	Higher power
Applications	Embedded systems	General-purpose
Development	Integrated development environment (IDE) provided by manufacturers, with specialized programming languages and tools	Standard development tools and languages such as C, C++, and assembly
Clock speed	Lower clock speed, typically less than 100 MHz	Higher clock speed, typically greater than 1 GHz