Embedded Systems CSEN701

Dr. Catherine Elias

Eng. Abdalla Mohamed

Office: C1.211
Mail : abdalla.abdalla@guc.edu.eg

Eng. Mohamed Elshafie

Office: C1.211
Mail : Mohamed.el-shafei@guc.edu.eg

Eng. Maysarah El Tamalawy

Office: C7.201
Mail : Maysarah.mohamed@guc.edu.eg

Outline :

- **Recap.**
- Sensors
- ADC
- Actuators
- PWM
- Examples

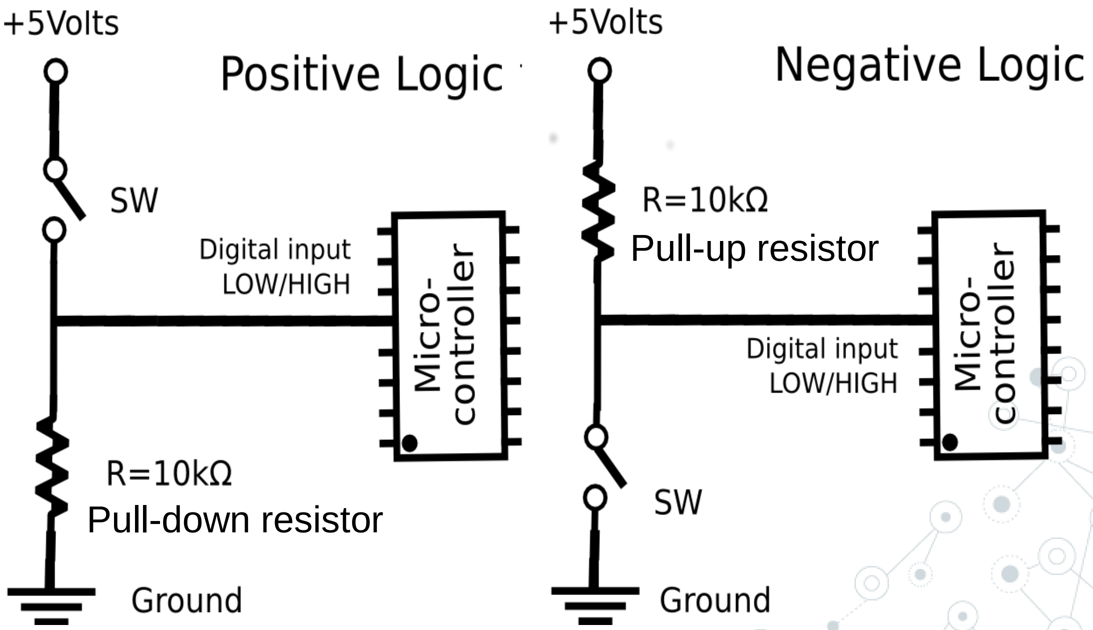
EX1

Implement an embedded C code to :

1. Connect push button A to pin 5 in PORT C (Positive Logic)
2. Connect push button B to pin 3 in PORT B (Negative Logic)
3. Apply the Internal pullup resistor to pin3 PORT B
4. Configure PIN 2 in PORTD as output
5. Connect Pin 2 to RED LED Pin5-- RED (Hardware step)
6. Turn on Red LED when A is pressed
7. Turn off the RED LED when B is pressed .

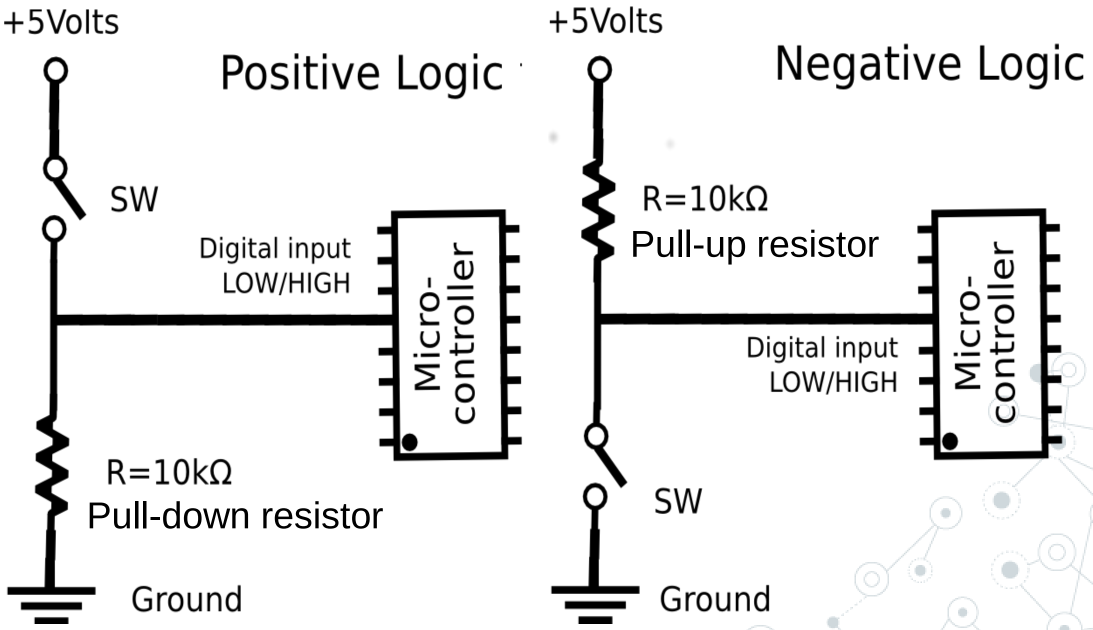
But first we have to understand different digital logics …

- Positive logic is the **default logic** , the input is initially low until the button is ON /activated so it deliver High voltage (5v) to the Microcontroller/LED when pressed.
- Sensors/Pins working with positive logic are called Active-HIGH.
- Pull-down resistors are associated with positive logic to **pull** the initial pin value **down** to GND thus preventing the floating of the input value.
- Button Is OFF/open -- Input = GND (0V)
- Button is ON/Closed -- Input = VCC (5V)



But first we have to understand different digital logics …

- Negative Logic connection operates in an opposite manner, the input is initially high until the button is On/activated it delivers GND (0V) to the Microcontroller/LED.
- Sensors/Pins working with negative logic are called Active-Low.
- Pull-up resistors are associated with negative logic to **pull** the initial pin value **up** to High voltage (5V) thus preventing the floating of the input value .
- Button Is OFF/open -- Input = VCC (5V)
- Button is ON/Closed -- Input = GND (0V)



EX1

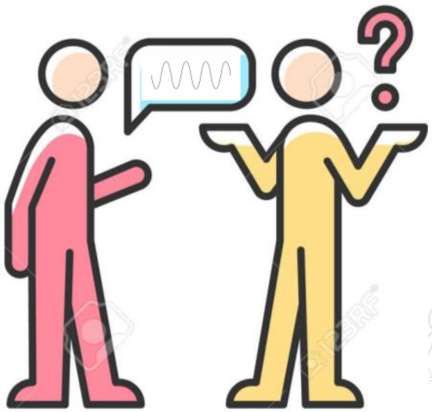
```
#include <avr/io.h>
int main (void){
DDRB = 0x00 ; DDRD = 0x00 ; DDRC = 0x00 ; PORTC = 0x00 ; PORTB=0x00; PORTD=0x00 ;    // initialize the registers
DDRC &=~(1<<5)  ;  // configure pin5 as input  in PORTA ( pushbutton A is connected to PIN 5 in positive Logic )
DDRB &=~(1<<3)  ;  // configure pin3 as input in PORTB ( pushbutton B is connected to PIN 3 in negative Logic )
PORTB |= (1<<3) ; // set bit 3 to HIGH to activate the internal pull-up resistor at pin 3
DDRD   |= (1<<2) ; // configure pin 2 as an output pin  at PORTD
while (1 ) {
if ( PINC & ( 1<<5) ) {   //  HINT : (PINC & 0b00100000) is only true when bit 5 at PINC is 1 (pushbutton A is pressed +ve L)
PORTD |= (1<<2) ;  // set the output to HIGH  to TURN ON the LED
}
if ( !(PINB & (1<<3) ) {   // (PINB & (1<<3)) is true when Bit 3 is ON ( not pressed ) so it will be false (!) if  pressed ( -ve L)
      PORTD &= ~(1<<2)  ; // set the output to LOW  by clearing bit 2        // pushbutton B is connected in negative logic
  } }                                              }
```

Dr.Catherine Elias
Eng. Abdalla Mohamed Eng. Mohamed Elshafie Eng. Maysarah El Tamalawy

Outline :

- Recap.
- **Sensors**
- ADC
- Actuators
- PWM
- Examples

Different physical changes happen in the environment around our system, and the system needs to measure these changes to respond to them.
BUT …
The system cannot understand the language the environment speaks



The environment *The system*

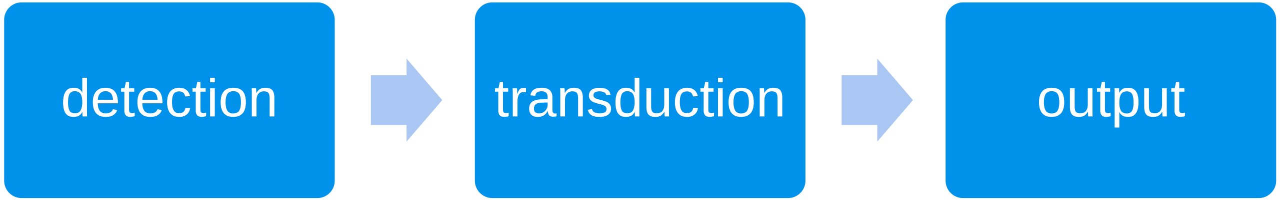
Sensors

That is why embedded system need sensors.
They are devices that detect or measure physical changes in the environment and convert them into electrical Signals or readable inputs to the system, to enable the system to respond to changes.

The environment   Sensor   *The system*

Sensors

The process undergoes 3 stages

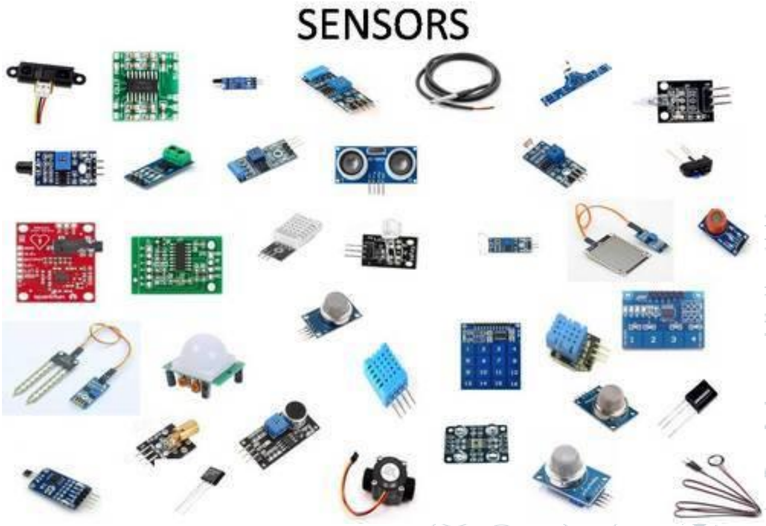


Detection

Sensors are designed to detect a physical phenomenon or a property

Examples

- Proximity sensor
- Temperature Sensor
- Infra-red sensor
- Light intensity sensor
- Microphone
- Pressure sensor
- Color sensor



Transduction

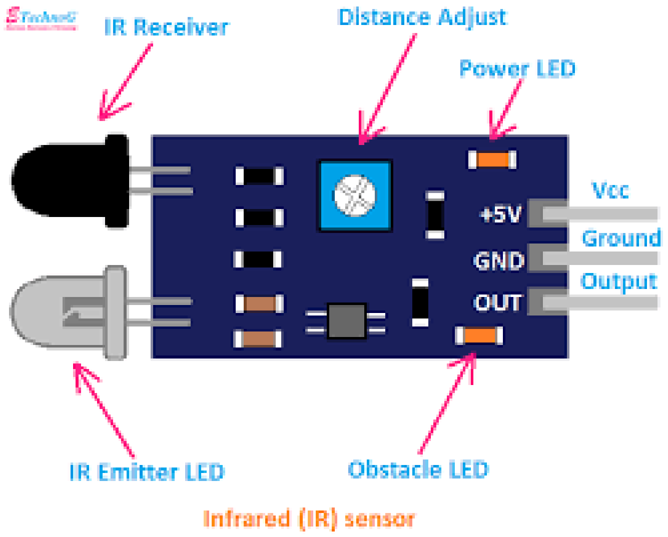
Conversion of the physical phenomenon into a measurable signal.
The measurable signal can be Vibrational (sound) , Thermal , optical, mechanical or any type of form / energy which can be eventually converted to **Electrical Output Signal** .

Examples

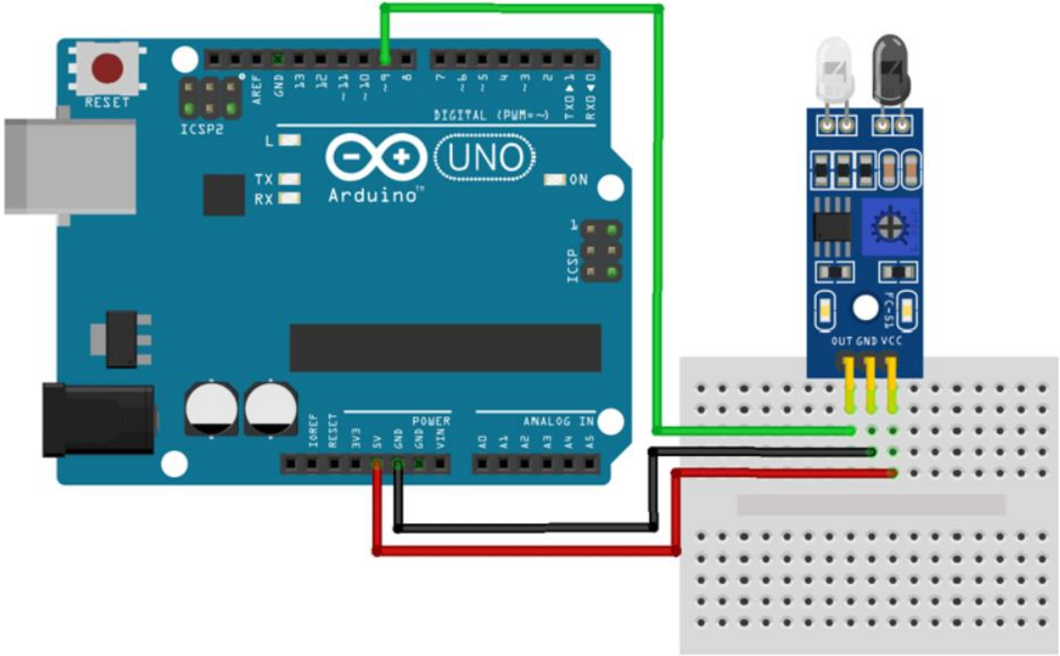
Sensor	functionality	from	to
Digital IR (infra-red sensor)	Detect presence of an object within a distance based on infra-red radiation	IR radiation	Digital electrical signal
Temperature	Measures amount of heat energy	Heat energy	Analogue electrical Signal
Ultrasonic	Measures distance/presence of target object	vibrations	Analogue/digital electrical signals
Light Intensity	Measures Light intensity	Light energy	Analogue electrical signals
Sound / Microphone	Measures sound level	Sound vibrations	Analogue electrical signals

Let’s get practical !

Let’s test a digital IR sensor, if the central IR sensor is over a black line / object , turn on the built in LED



ARDUINO UNO (AVR- based architecture)



```
int IRSensor = 9; // connect IR sensor module to Arduino pin D9
int LED = 13; // connect LED to Arduino pin 13
void setup(){
  Serial.begin(115200); // Init Serial at 115200 Baud Rate.
  pinMode(IRSensor, INPUT); // IR Sensor pin INPUT
  pinMode(LED, OUTPUT); // LED Pin Output
}
void loop(){
  int sensorStatus = digitalRead(IRSensor);
  if (sensorStatus == 1) // Check if the pin high or not
  {
    // if the pin is high turn on the onboard Led
    digitalWrite(LED, HIGH); // LED HIGH
    Serial.println("Motion Detected!"); // print Motion Detected! on the serial monitor window
  }
  else  {
    //else turn low the onboard LED
    digitalWrite(LED, LOW); // LED LOW
    Serial.println("Motion Ended!"); // print Motion Ended! on the serial monitor window
  }
}
```

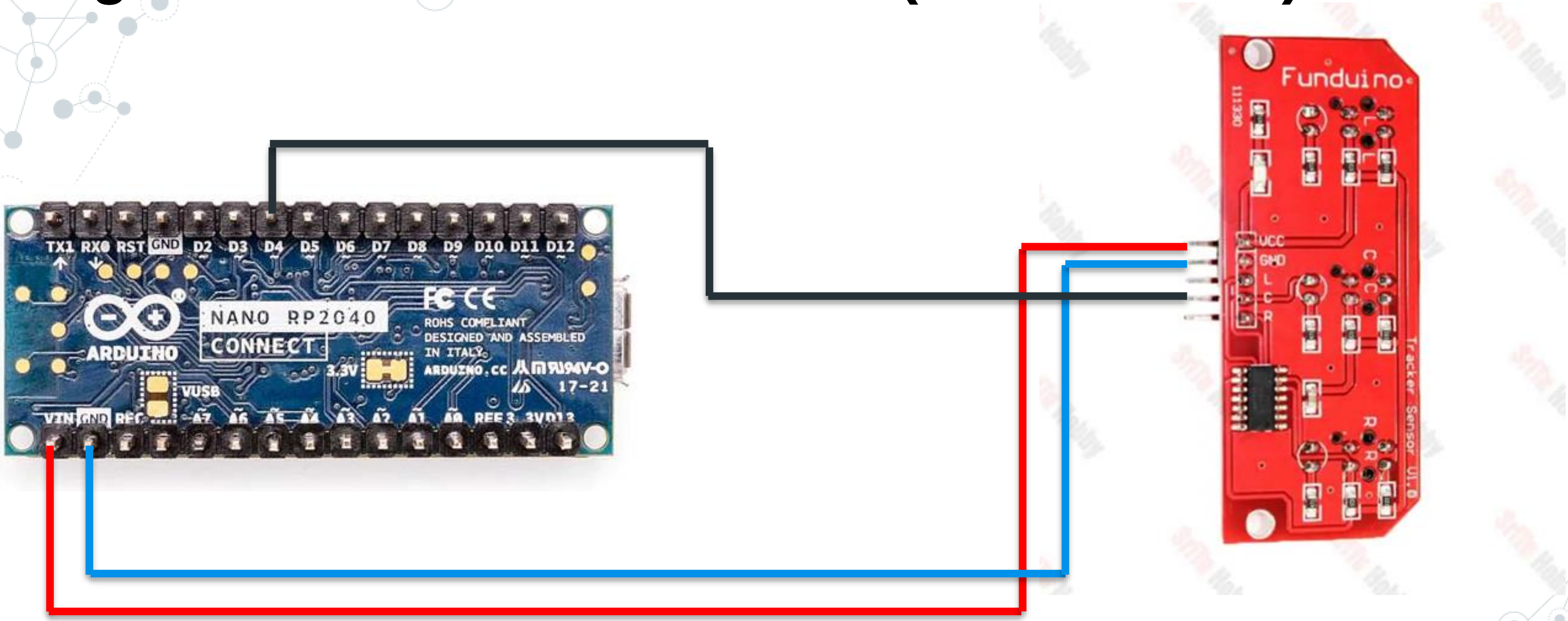
Functions used in AVR :

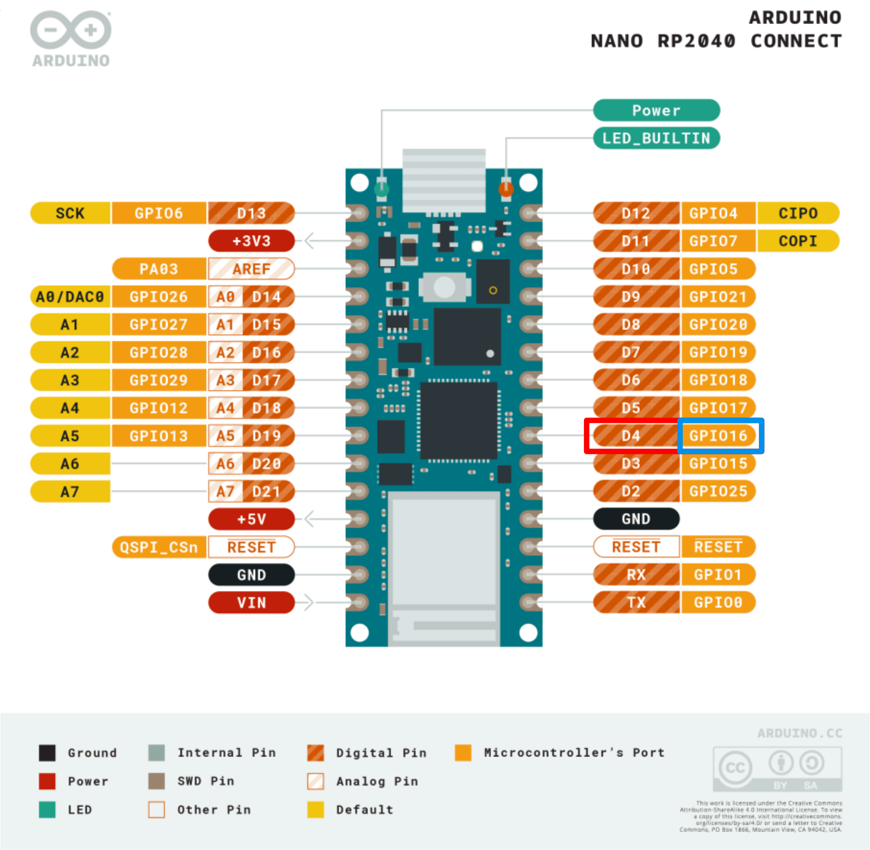
- pinMode (pin, direction either OUTPUT or INPUT) set pin direction
- digitalWrite (pin, value) write a digital pin either HIGH or LOW
- analogWrite (pin, value from 0 to 255) // write a value from 0 to 255 to a PWM pin . (0 to 5 volts)
- digitalRead (pin) read digital input HIGH or LOW
- analogRead (pin) // read analogue input value from 0 to 1023

Serial.begin (); // start the serial monitor

Serial.println (" "); //print on the serial monitor

Wiring On Arduino Nano RP2040 (ARM based)




ARDUINO
ARDUINO
NANO RP2040 CONNECT
Power
LED_BUILTIN
SCK GPIO6 D13
+3V3
PA03 AREF
A0/DAC0 GPIO26 A0 D14
A1 GPIO27 A1 D15
A2 GPIO28 A2 D16
A3 GPIO29 A3 D17
A4 GPIO12 A4 D18
A5 GPIO13 A5 D19
A6 A6 D20
A7 A7 D21
+5V
QSPI_CSn RESET
GND
VIN
D12 GPIO4 CIPO
D11 GPIO7 COPI
D10 GPIO5
D9 GPIO21
D8 GPIO20
D7 GPIO19
D6 GPIO18
D5 GPIO17
D4 GPIO16
D3 GPIO15
D2 GPIO25
GND
RESET RESET
RX GPIO1
TX GPIO0
Ground
Power
LED
Internal Pin
SWD Pin
Other Pin
Digital Pin
Analog Pin
Default
Microcontroller's Port
ARDUINO.CC

```c
#include <stdio.h>
#include "pico/stdlib.h"
int main() {

   const uint led_pin = 6;
   const uint IR_pin = 16;


   // Initialize LED and IR pin
   gpio_init(led_pin);
   gpio_init(IR_pin);

   gpio_set_dir(led_pin, GPIO_OUT);
   gpio_set_dir(IR_pin, GPIO_IN);

   // Initialize chosen serial port
   stdio_init_all();

   // Loop forever
   while (true) {

      // if IR is ON turn on the led
      if(gpio_get (IR_PIN))}
        gpio_put(led_pin, true);
      }
      else {
       gpio_put(led_pin,false); }}}
```

Pico C SDK

- Introduction
- Hardware APIs
- High Level APIs
- Third-party Libraries
- Networking Libraries
- Runtime Infrastructure
- External API Headers

- `void gpio_init (uint gpio)`

  Initialise a GPIO for (enabled I/O and set func to GPIO_FUNC_SIO)

- `void gpio_deinit (uint gpio)`

  Resets a GPIO back to the NULL function, i.e. disables it.

- `void gpio_init_mask (uint gpio_mask)`

  Initialise multiple GPIOs (enabled I/O and set func to GPIO_FUNC_SIO)

- `static bool gpio_get (uint gpio)`

  Get state of a single specified GPIO.

- `static uint32_t gpio_get_all (void)`

  Get raw value of all GPIOs.

- `static void gpio_set_mask (uint32_t mask)`

  Drive high every GPIO appearing in mask.

- `static void gpio_clr_mask (uint32_t mask)`

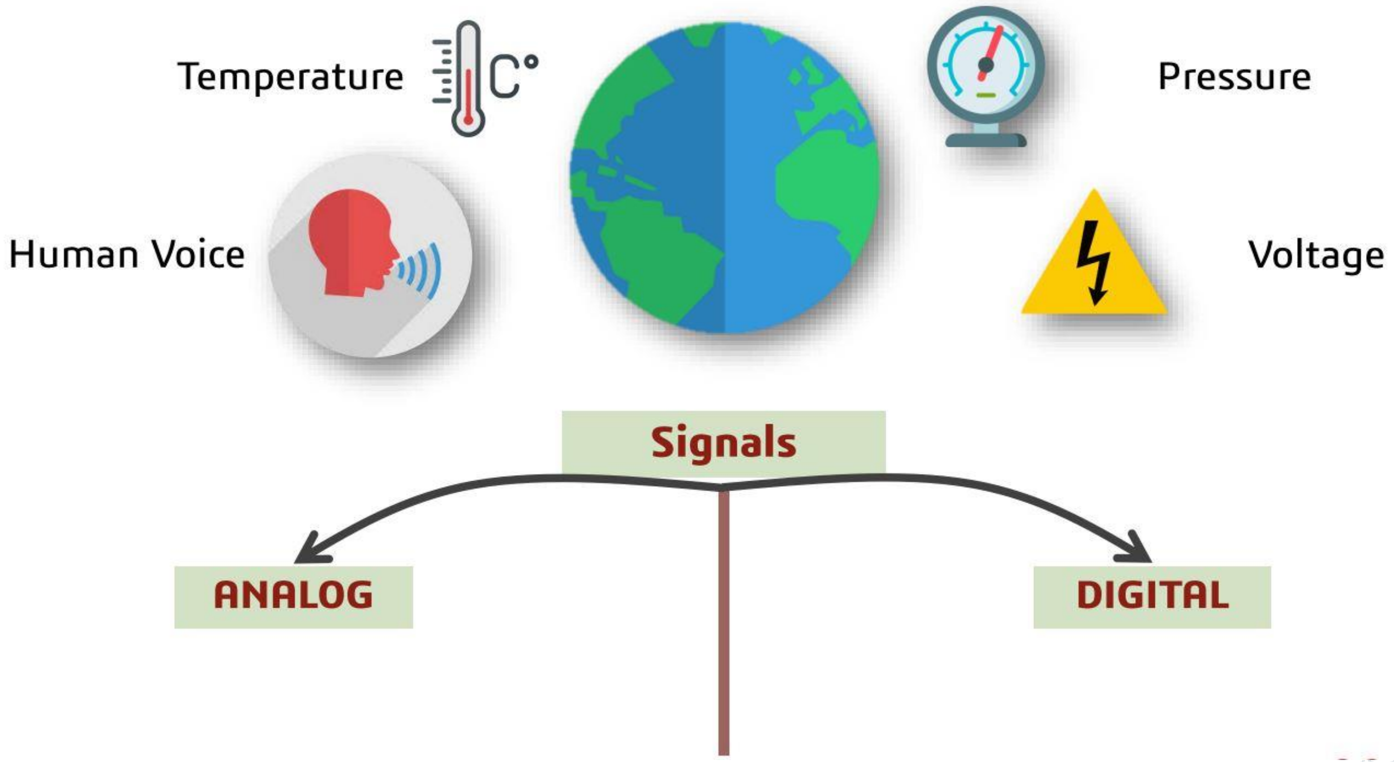
  Drive low every GPIO appearing in mask.

- `static void gpio_xor_mask (uint32_t mask)`

Outline :

- Recap.
- Sensors
- **ADC**
- Actuators
- PWM
- Examples

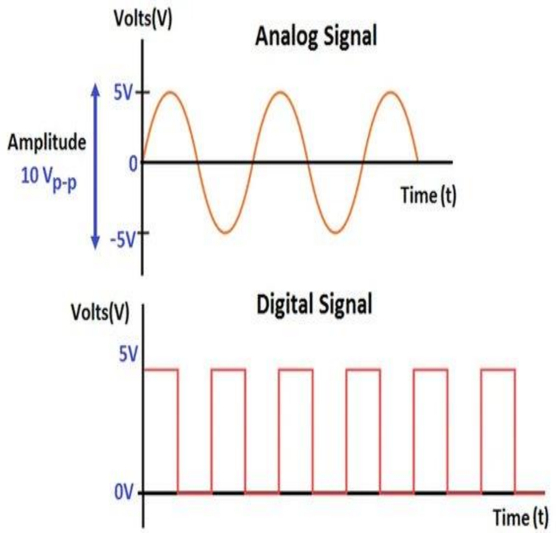
Real World Data



Output Signal

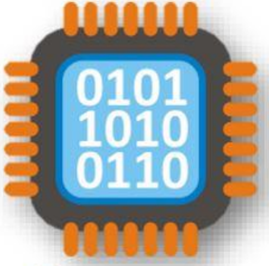
<u>The electrical output signal can be either Analogue or digital signals depending on the sensor :</u>

- An analog signal is a continuous representation of a physical quantity that can vary smoothly over time. It is characterized by an infinite number of possible values within a given range . An electrical analogue signal can have any voltage value from (0 to 5 V) for example .
- A digital signal is a discrete representation either 0 (low) or 1 (HIGH) perfect as an input for binary systems as Microcontrollers . An electrical digital signal output from an Active-High digital sensor can 5V (On state) or 0 (Off state) V for example.



Analog Signals	Digital Signals
Continuous and smooth waveform	Discrete and stepped waveform
Represents real-world data as a continuously varying voltage or current .Range of values (0 to 5 V).	Represents data as discrete values (0s and 1s). Either 0 (LOW) or 1 (HIGH) .
Output is an analog voltage or current directly proportional to the measured quantity	Output is a binary representation (0 or 1) of the measured quantity
Infinite resolution, theoretically	Limited by the number of bits in the ADC (e.g., 8-bit, 10-bit, 12-bit)
Requires specialized analog processing circuitry (filters, amplifiers)	Easily processed using digital logic
Prone to signal degradation during transmission and storage	Less susceptible to degradation; easier to transmit and store

Microcontroller will receive **analog signals**

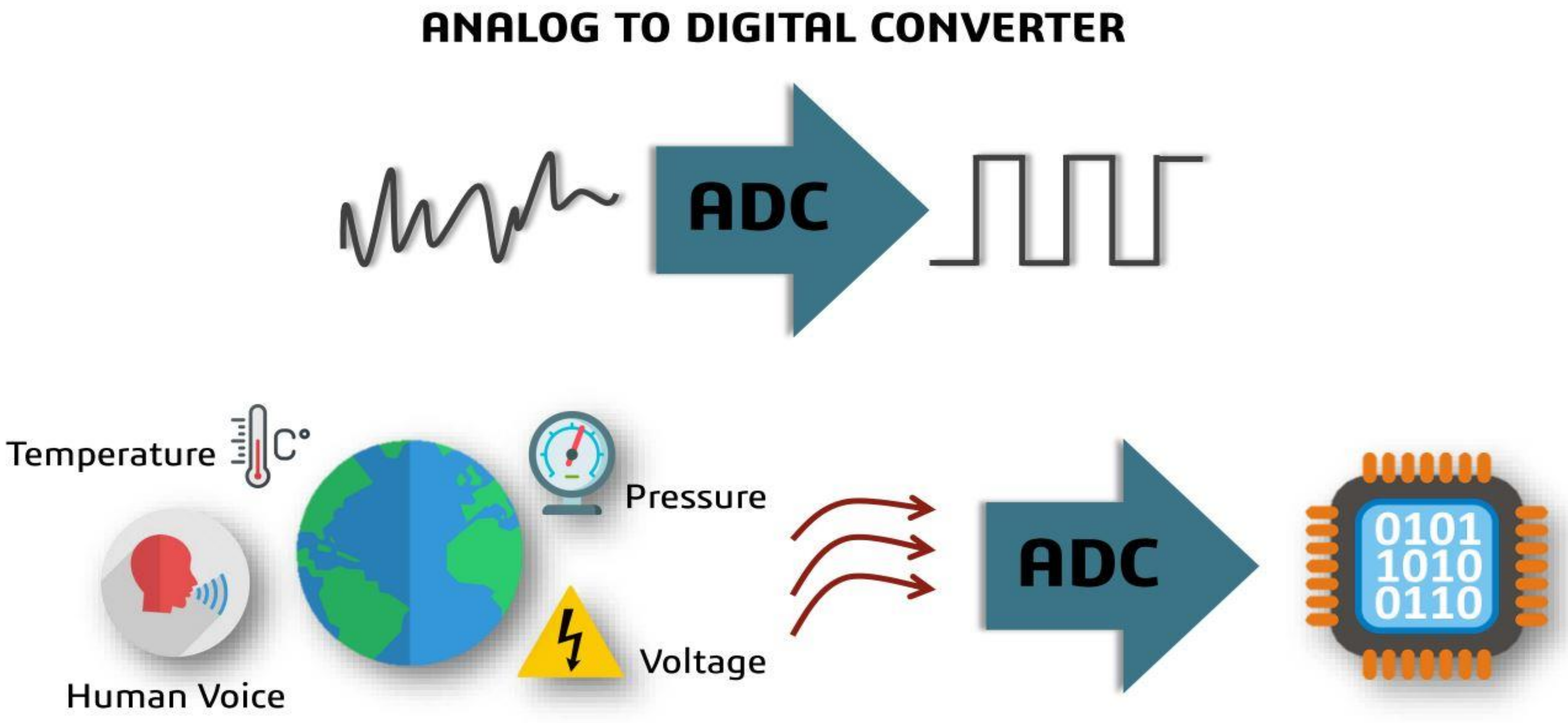


SO

What if we need to get some non-digital data in the microcontroller ?

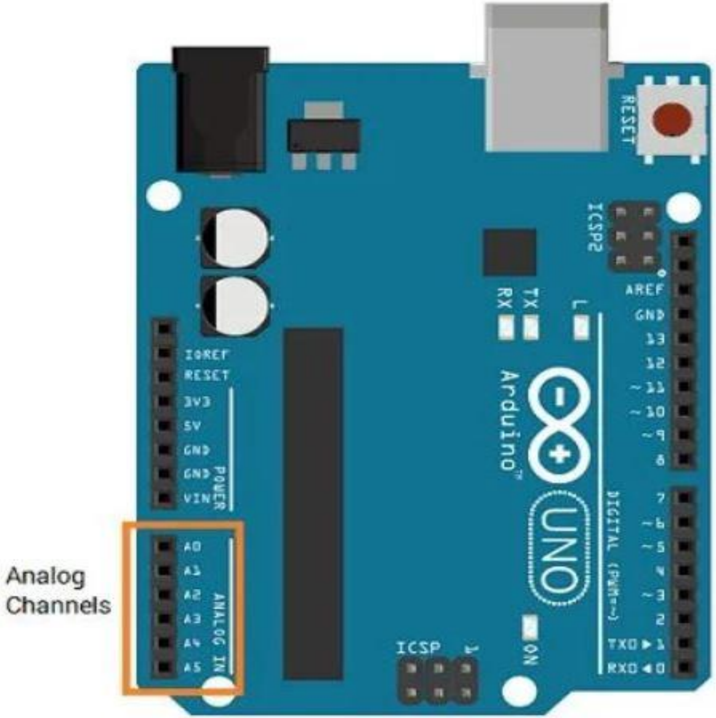
but

The Microcontroller have to deal with digital information "They only understand '0' or '1' values


ANALOG TO DIGITAL CONVERTER
ADC
Temperature
C°
Pressure
Human Voice
Voltage
ADC
0101
1010
0110

ADC Pins of Arduino Uno

- **Arduino Uno has 6 0n-board ADC channels** which can be used to read analog signal in the range 0-5V.
- It has 10-bit ADC means it will give digital value in the **range of 0 – 1023 (2^10).** This is called as a resolution which indicates the number of discrete values it can produce over the range of analog values.



Arduino ADC pin Diagram

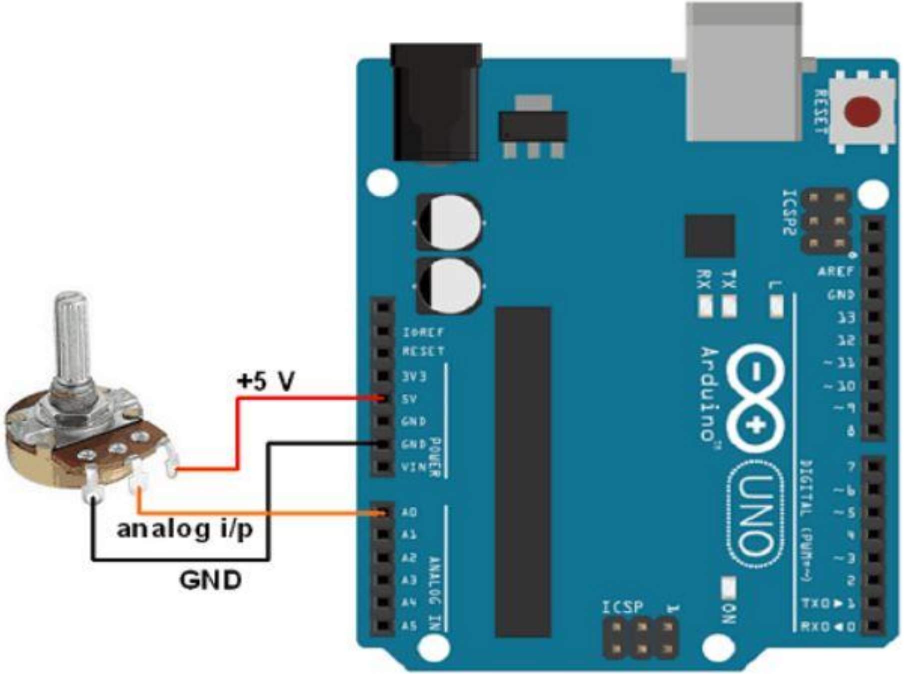
Digital Output value Calculation:

- ADC Resolution = Vref / ((2^n) - 1)
- Digital Output = Vin / Resolution.

Vref - The reference voltage is the maximum value that the ADC can convert to keep things simple, let us consider that Vref is 5V,

- For 0 Vin, digital o/p value = 0
- For 2.5 Vin, digital o/p value = 512 (10-bit)
- For 5 Vin, digital o/p value = 1023 (10-bit)

Potentiometer Interfacing with Arduino Uno



Potentiometer connected Arduino ADC Channel

```
int sensorPin = A0; // input pin for the potentiometer
int digitalValue = 0;// variable to store the value coming from the sensor
void setup() {
Serial.begin(9600); }
void loop() {
digitalValue = analogRead(sensorPin);// read the value from the analog channel
Serial.print("digital value = ");
 Serial.println(digitalValue); //print digital value on serial monitor
 delay(1000);
}
```

COM4 (Arduino/Genuino Uno)

Send

```
digital value = 0
digital value = 0
digital value = 0
digital value = 0
digital value = 42
digital value = 84
digital value = 202
digital value = 303
digital value = 352
digital value = 461
digital value = 502
digital value = 512
digital value = 623
digital value = 754
digital value = 845
digital value = 965
digital value = 992
digital value = 1018
digital value = 1021
digital value = 1016
digital value = 1021
digital value = 1019
```

Autoscroll | No line ending | 9600 baud

Outline :

- Recap.
- Sensors
- ADC
- **Actuators**
- PWM
- Examples

Actuators

After the system takes in information (physical changes) through sensors, it needs to respond somehow (or Affect the environment it is in) . The system produces outputs through devices called actuators . Any Output device can be considered as an actuator .

Examples :

- LCD display
- LEDs
- Servo Motors
- Dc Motors
- Stepper Motor
- Buzzer

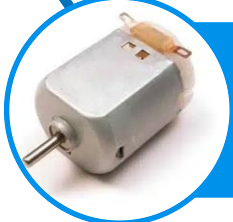
Actuators

They are devices that convert electrical energy into another form of physical energy that can interfere with the environment. Actuators can receive Analogue or digital signals depending on the actuator .

Electrical Energy

Physical energy
(motion)

Actuators Types



Digital

- Output values are only on/off (1/0)
- DC motor
- LEDs

Analog

- When the device needs to function at a range of values not only ON/OFF states
- Servo Motor , PWM-controlled DC motor/LEDs

Outline :

- Recap.
- Sensors
- ADC
- Actuators
- **PWM**
- Examples

What is PWM ?

Pulse Width Modulation is a technique used to control analog devices, using a digital signal. This technique can be used to output an analog-like signal from a digital device (microcontroller) . We can control motors, lights, actuators, and more using the generated PWM signal.

Applications :

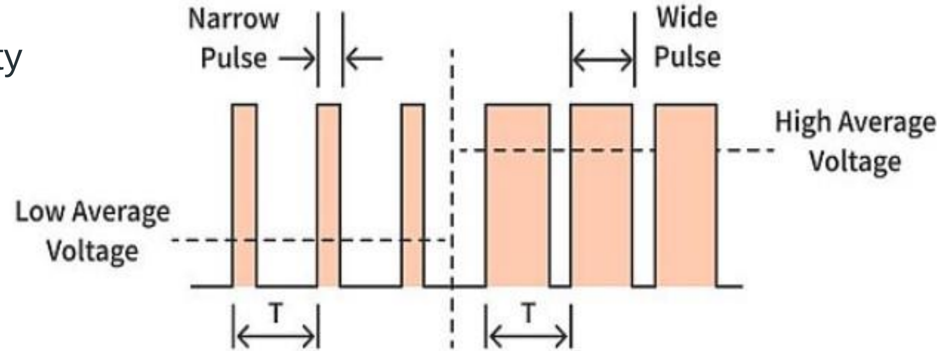
We can control the power delivered to electrical devices using Pulse Width Modulation (PWM) signals. Now, because of its high efficiency, low power loss, and its ability to precisely control the power, this technique is used in many applications like:

- Controlling the speed of DC motors and servo motors.
- Dimming of LEDs and soft-blinking. (Lights can go from full intensity to dark slowly and slowly raised to full intensity again using PWM)

But how ……..

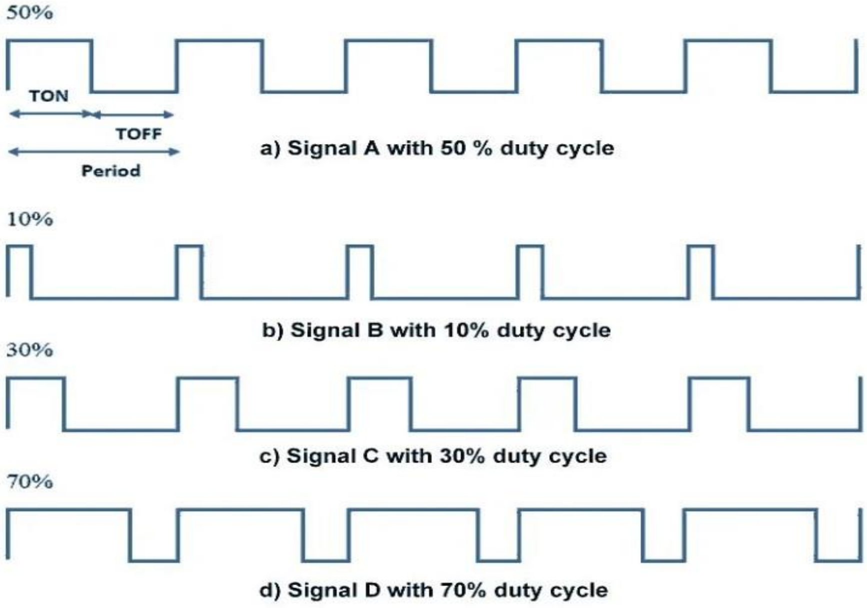
PWM is based on varying the width of digital Pulse with keeping the frequency constant, thus the average power delivered by the output is varied depending on what we call the **Duty cycle**. A period of a pulse consists of an **ON** cycle (5V HIGH) and an **OFF** cycle (0V LOW). The fraction for which the signal is ON over a period is known as the **duty cycle (D)** .

- Average voltage supplied is proportional to duty cycle .
- Duty cycle (D) = Time_ON (High Time) / Total_period_time(T_on+T_off)



But how ……..

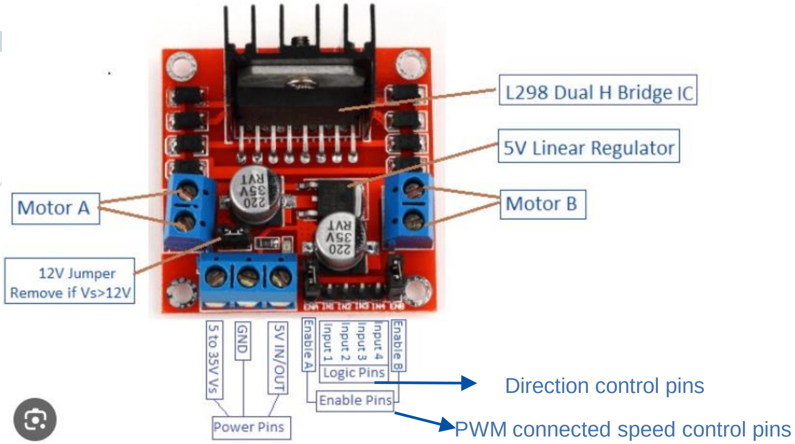
- Consider a pulse that is 10ms and the pulse is High for 5 ms then the duty cycle is D = (5/10)*100 = 50% .
- Using this switching ON&OFF technique the power delivered to the actuator is controlled .
- We can vary the speed of the motor by altering the duty cycle (increasing or decreasing the HIGH time) thus creating an analogue like signal form using digital pulses .

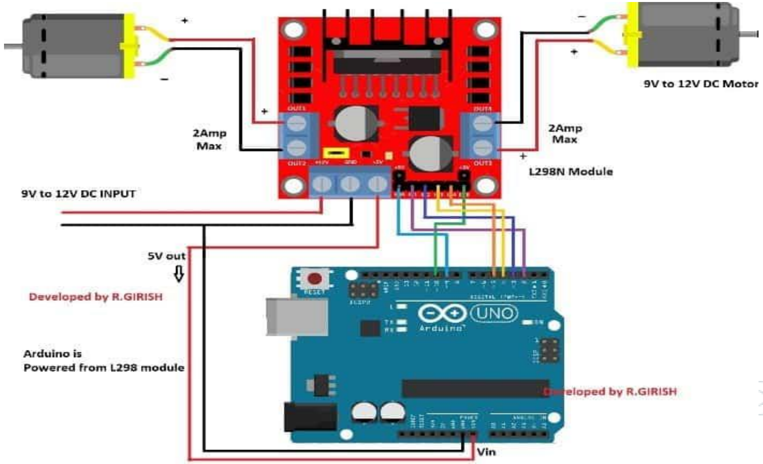


PWM in controlling Motors

Dc Motors speed can be controlled using PWM pins in the Arduino, however an interface circuit must be used. The interface circuit that is used to drive the motor is called H-Bridge.

<u>What is H-Bridge :</u> A H bridge is an electronic circuit configuration used to control the direction and speed of a motor . It is a mid-way circuit between the micro-controller and the motors .





Outline :

- Recap.
- Sensors
- ADC
- Actuators
- PWM
- **Examples**

Example

```
// defining pins for the first motor
int enA = 10;
int in1 = 9;
int in2 = 8;
void setup()
{
  // set all connected pins as output
  pinMode(enA, OUTPUT);
  pinMode(enB, OUTPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);
}
void loop()
{
  // turn the first motor on
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  analogWrite(enA, 200);
   delay(2000) ;   // turn on motor clockwise for 2 seconds at 200 speed max is 255
  digitalWrite(in1, LOW);
  digitalWrite(in4, HIGH);  //reverse the direction to anti-clockwise
  analogWrite(enA, 100);     // lower the speed to 100
  delay(2000); }
```

Functions used in AVR :

- pinMode(pin, direction either OUTPUT or INPUT) set pin direction
- digitalWrite(pin, value) write a digital pin either HIGH or LOW
- analogWrite(pin, value from 0 to 255) // write a value from 0 to 255 to a PWM pin . (0 to 5 volts)
- digitalRead(pin) read digital input HIGH or LOW
- analogRead(pin) // read analogue input value from 0 to 1023

  Serial.begin(); // start the serial monitor

  Serial.println(" "); //print on the serial monitor

THANK YOU