PROGRAMMING WITH ARDUINO

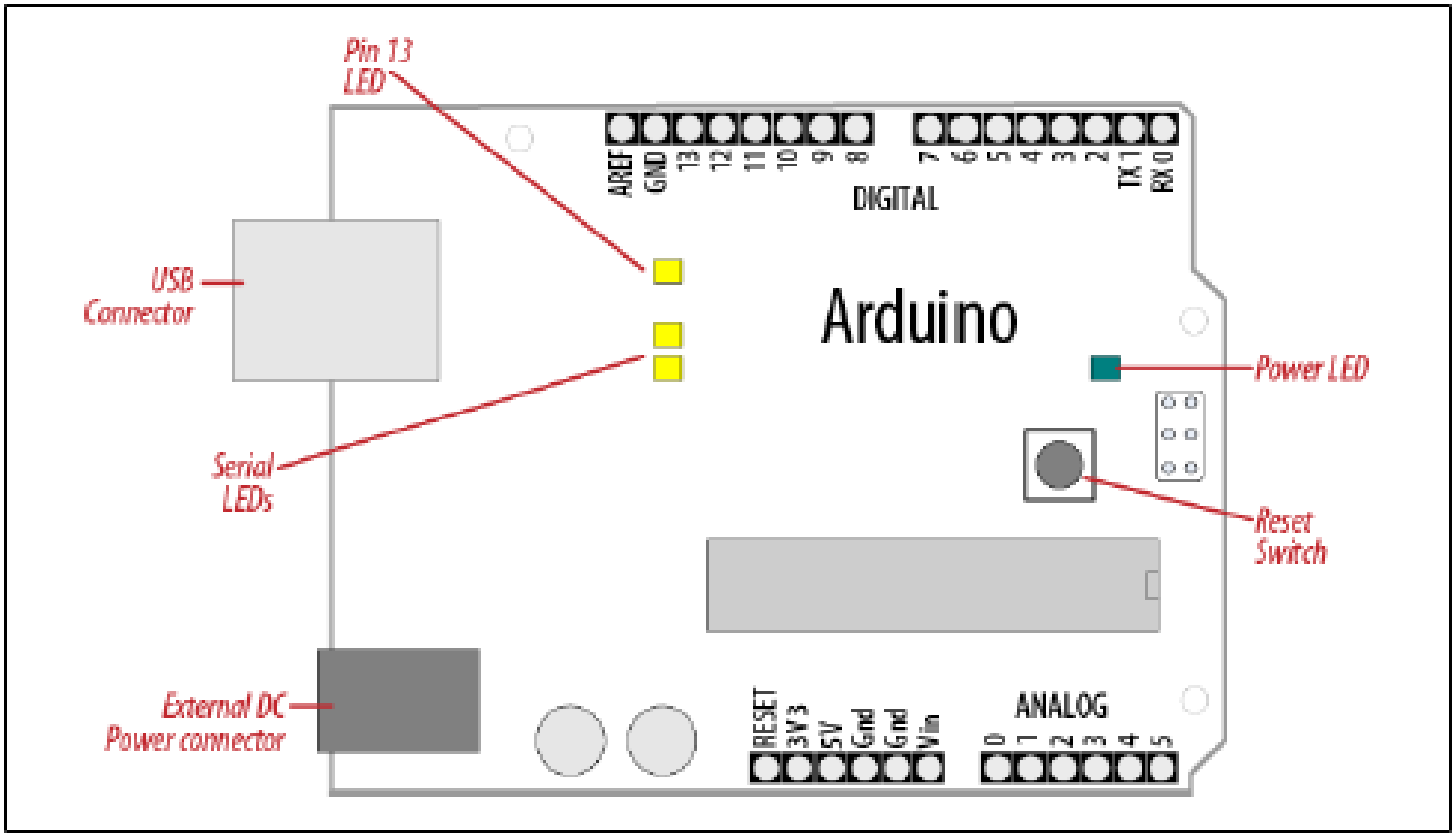
What is Arduino?

- An open-source hardware platform based on an Atmel AVR 8-bit microcontroller and a C++ based IDE
  - Arduino Due is based on a 32-bit ARM Cortex
- Arduino boards are able to read inputs - read a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED
- Over **300000** boards have been manufactured

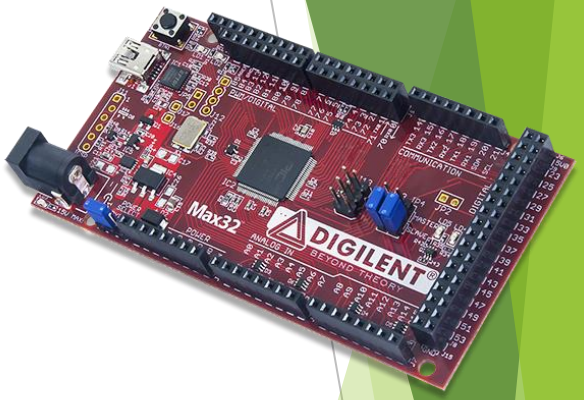
Why Arduino?

- Inexpensive (<$50)
  - microcontroller platforms
- Cross-platform
  - The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems & Online IDE
- Simple, clear programming environment
  - easy-to-use for beginners (C++ environment)
- Open source and **extensible software**
  - The language can be expanded through C++ libraries
- Open source and **extensible hardware**
  - circuit designers can make their own version of the module, extending it and improving it

Typical Arduino Board

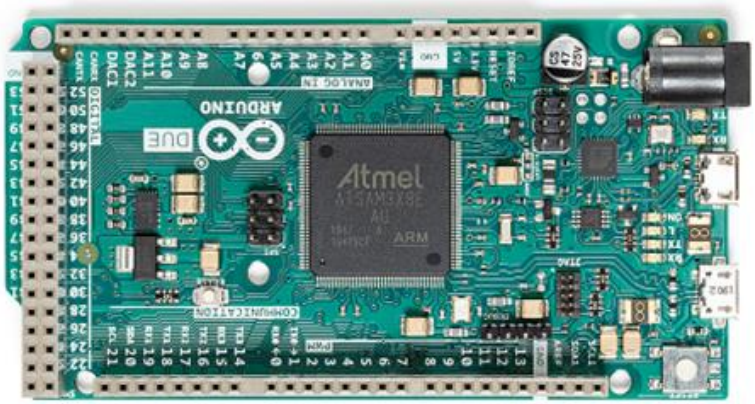


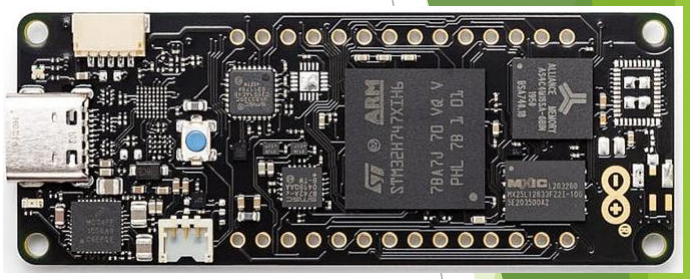
Arduino Boards



	Arduino Nano v3 / Arduino UNO	Arduino Max32
Επεξεργαστής	ATmega328 (16MHz)	Microchip PIC32MX795F512 (80MHz)
Μνήμη	32KB Flash 2KB SRAM	512KB Flash 128KB RAM

Arduino Boards (ARM-based)





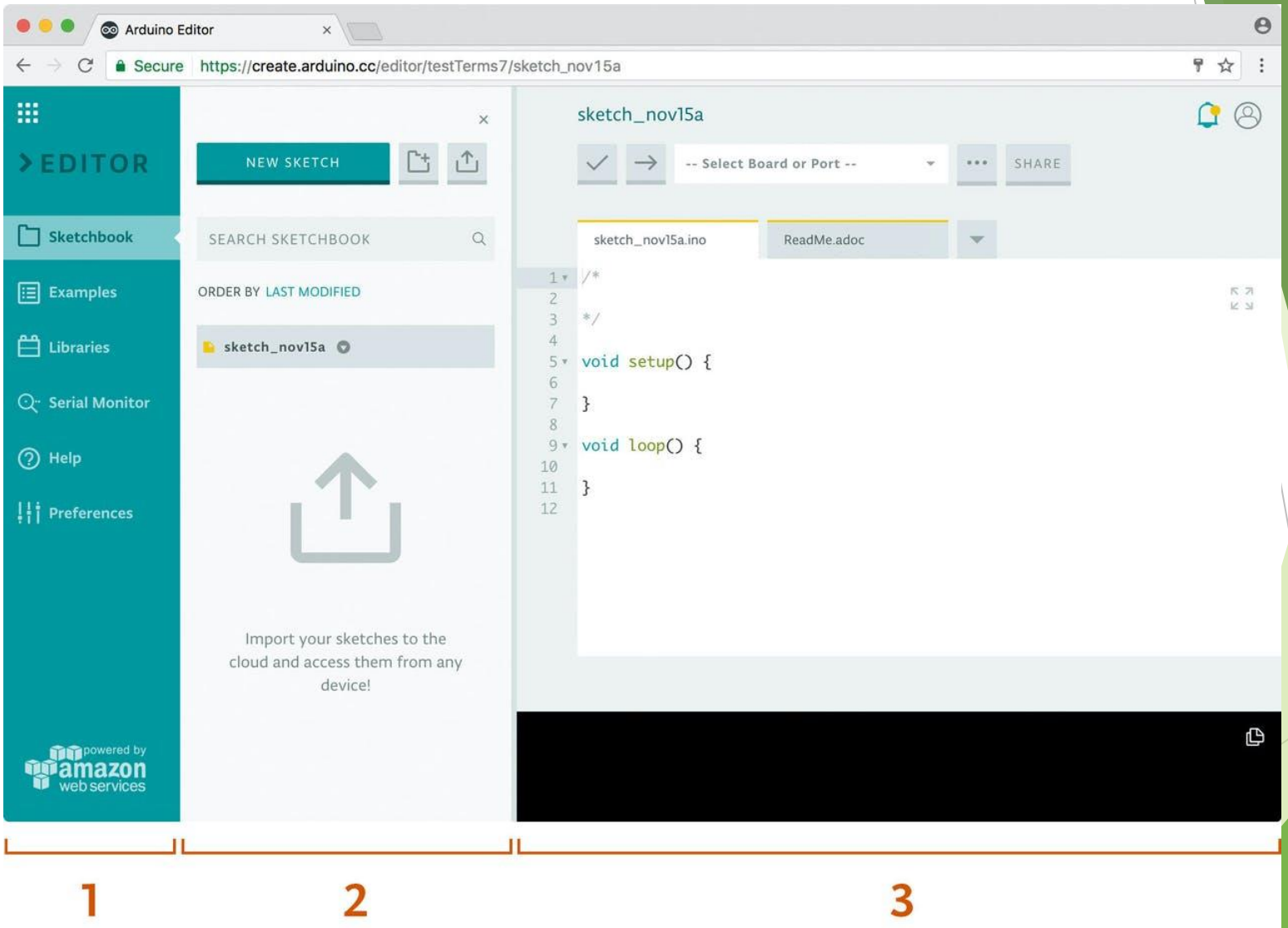
	Arduino Due	Arduino Portenta H7
Επεξεργαστής	32-bit ARM Cortex-M3 CPU (84MHz)	STM32H747 dual core Cortex M7@480 MHz + Cortex M4@240MHz
Μνήμη	512KB Flash 96KB SRAM	16 MB Flash 8 MB SDRAM

https://en.wikipedia.org/wiki/List_of_Arduino_boards_and_compatible_systems

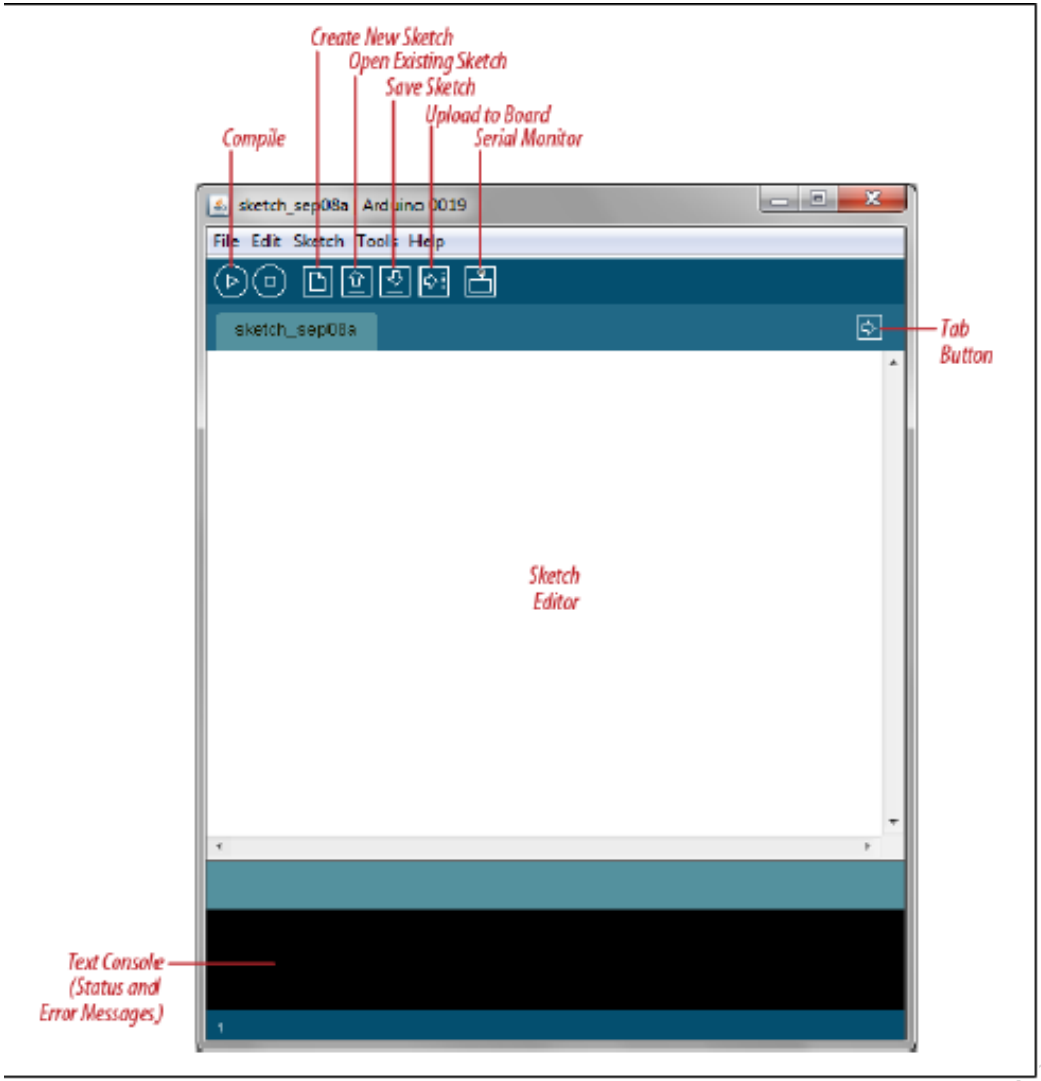
Arduino Software (IDE)

- The Arduino Software (IDE) allows you to **write programs** and **upload them to your board**
- In the Arduino Software page you will **find two options:**
  - online IDE (Arduino Web Editor)
    - It will allow you to **save your sketches in the cloud**, having them **available from any device and backed up.**
    - You will always have the **most up-to-date version** of the IDE without the need to install updates or community generated libraries.
  - desktop IDE
    - If you would rather work **offline**

Arduino Online IDE



Arduino Desktop IDE



OUTLINE

- Biomedical Projects using Arduino
- Essential Programming Concepts
  - Delay
  - Infinite Loop
- Sensors & Actuators
- High-Level Language Extensions
- Timers and Internal Interrupts
- Connecting through GPS & GSM
- Finite State Machine (FSM)
- More Complex Arduino

Biomedical Projects using Arduino

https://create.arduino.cc/projecthub/projects/tags/arduino
https://create.arduino.cc/projecthub/search?q=medical

Health Band - A Smart Assistant for the Elderly (1/3)

- This health band can assist old people in their daily lives, leaving the family stress free!
- «A virtual assistant that could monitor my grandma's behavior and activities, and if needed could also alert a family member in case of an emergency.»

- https://create.arduino.cc/projecthub/Technovation/health-band-a-smart-assistant-for-the-elderly-0fed12

Health Band - A Smart Assistant for the Elderly (2/3)

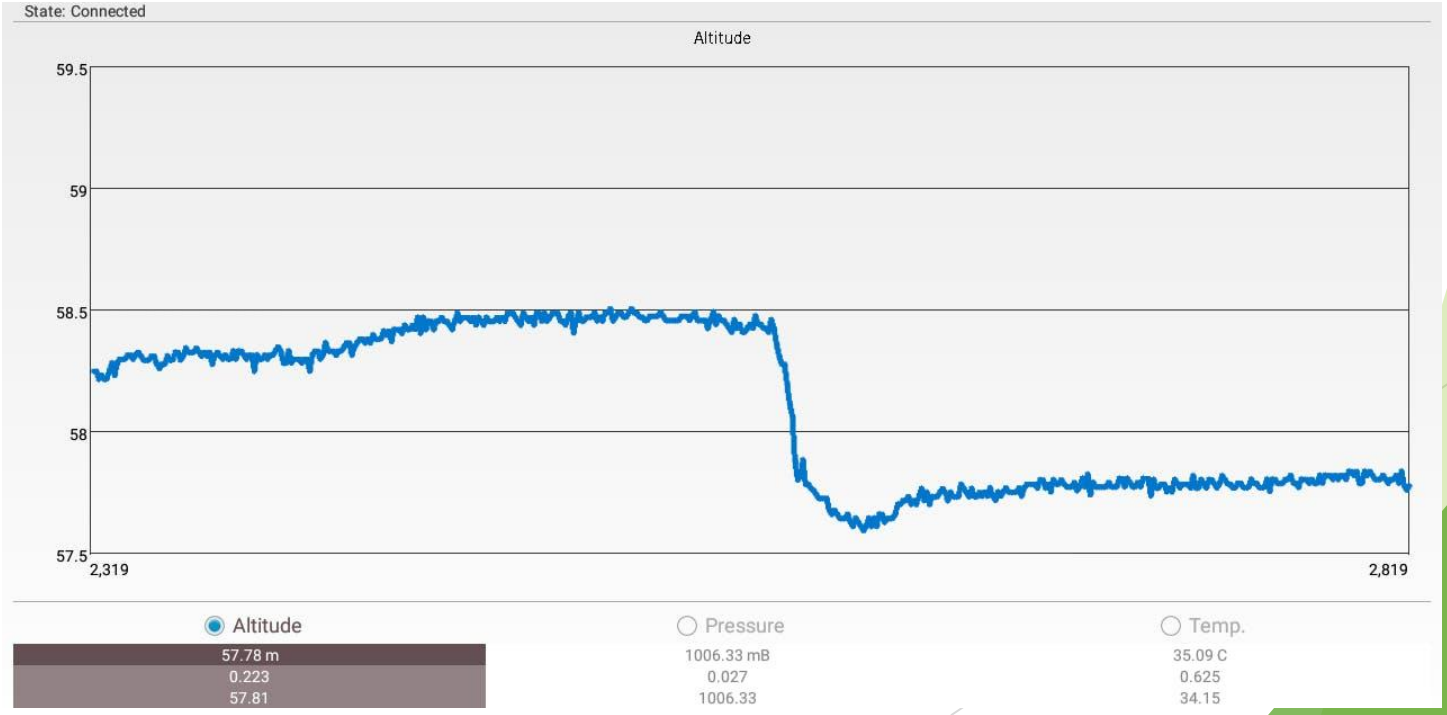
Comfortable and Compact wearable Technology

Data clearly presented on Blynk App On Smartphone

- In the heart of our system we have the **Arduino Nano** which is then connected to the **DPS310 pressure sensor** via the I2C bus.
- **Actions and Behaviors** will be coded to simulate the live motion of the patient. Using these graphs or values we determine the state of the patient and alert a member in case of an emergency.

Health Band - A Smart Assistant for the Elderly (3/3)

- Here are the main functions and movements that our band will be able to analyse/detect:
  - **Fall** - studies show that the biggest issue with elders are of them losing balance and falling down.



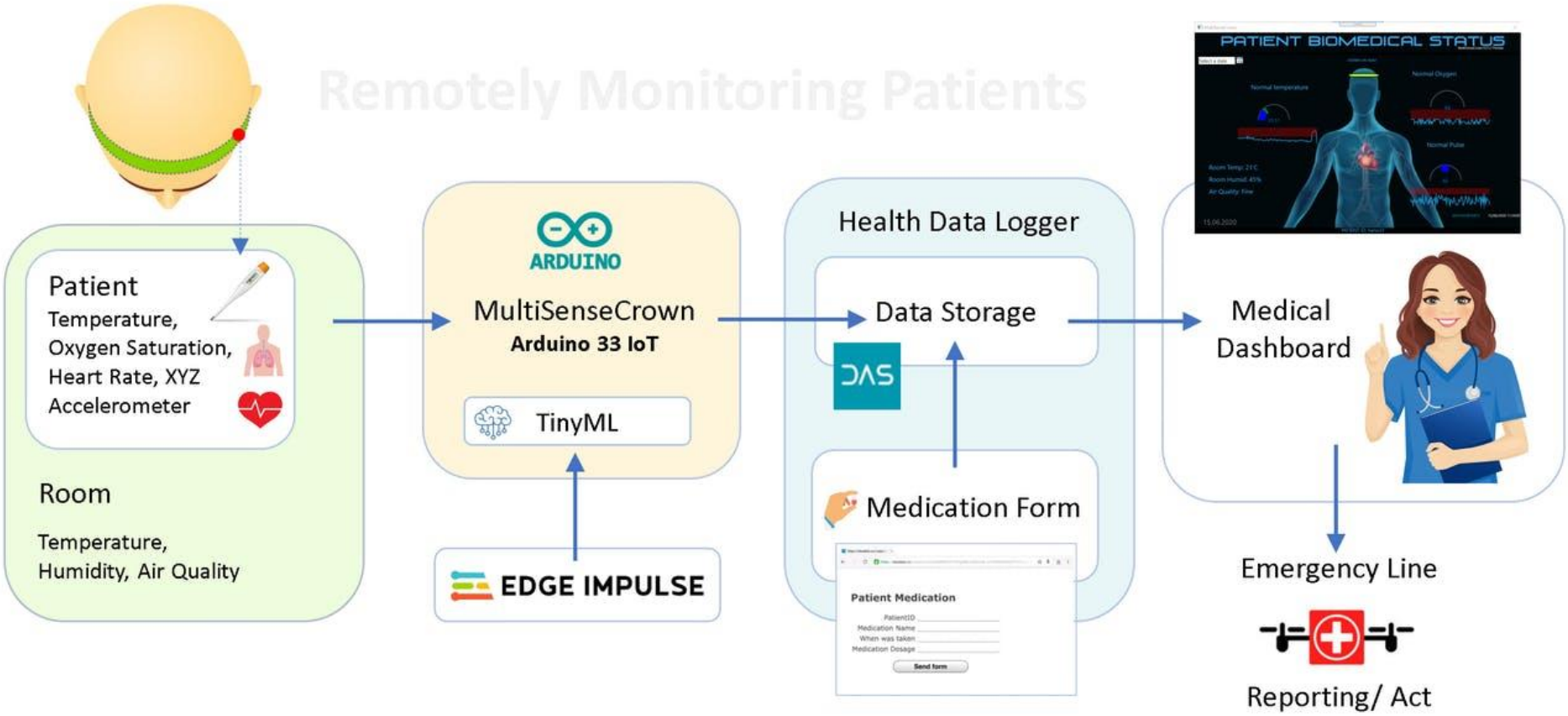
MultiSenseCrown (1/5)

- Advanced but yet affordable end-to-end **medical remote monitoring solution** of patient vitals and influence by the indoor air quality.

- **System Architecture**
  - **MultiSenseCrown** is a proposal of telemedicine usecase while offering an all-in-one complete monitoring system for human vitals but also environmental conditions, follow the system architecture below.

- https://create.arduino.cc/projecthub/dasdata/multisensecrown-e0daf9

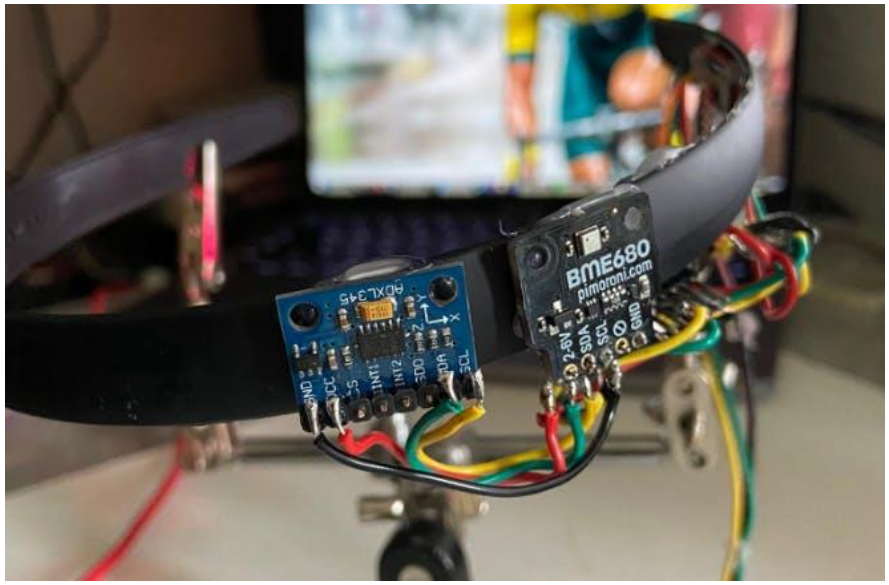
MultiSenseCrown (2/5)



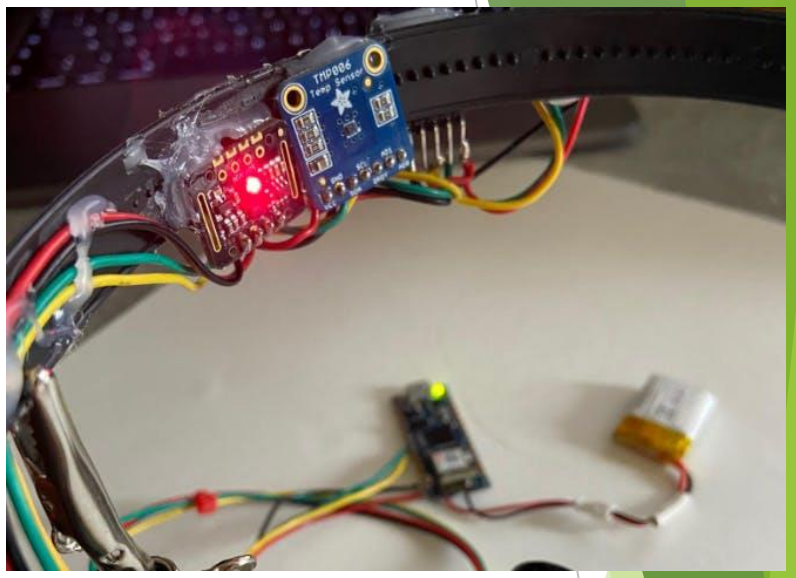
- **Human Vitals** can be easily measured using sensors such as MAX30102 (Oxygen Saturation, Pulse) or TMP006 for Contactless Infrared Temperature
- **Indoor environmental** conditions will be measured by the BME680 from BOSCH (relative humidity, barometric pressure, ambient temperature and gas (VOC)

https://www.arduino.cc/reference/en/libraries/harvard_tinymlx/

MultiSenseCrown (3/5)



Front View



Inside View

- **Attach Battery:** This wearable device should be also mobile, therefore we should add a battery.

MultiSenseCrown (4/5)

dasdata.co

DASDATA | My Data | User Account

md / MultiSenseCrown - Patient Vital Monitoring 2122 records

Last 200 Entries | Last 2 Days | Notebook | Clone

id	AIDate	TemperatureBody	OxygenSaturation	HeartRate	AccelX	AccelY	AccelZ	Medication_Note	IndoorTemperatureRAW	IndoorHumidRAW	IndoorPressureRAW	IndoorGasRAW
3775753	15.06.2020 12:16:25	35.52	96	98	-2.79	8.83	4.12	nano33	3237	100000	64897	9332177
3775752	15.06.2020 12:16:20	35.69	95	86	-2.86	8.79	4.20	nano33	3108	100000	152798	9332177
3775751	15.06.2020 12:16:16	35.74	95	90	-2.86	8.79	4.24	nano33	3237	100000	64897	177
3775750	15.06.2020 12:16:12	35.87	95	97	-2.90	8.59	4.47	nano33	3237	100000	64897	806516
3775749	15.06.2020 12:16:08	35.92	96	97	-2.90	8.43	4.59	nano33	3108	100000	152798	9332177
3775748	15.06.2020 12:16:04	35.83	94	105	-2.47	8.28	5.02	nano33	3237	100000	64897	9332177
3775747	15.06.2020 12:15:59	36.16	94	67	-1.02	8.28	5.06	nano33	3108	100000	152798	9332177
3775746	15.06.2020 12:15:55	36.06	94	99	-3.26	8.51	4.20	nano33	3237	100000	64897	806516
3775745	15.06.2020 12:15:51	35.84	94	91	-3.41	8.43	4.28	nano33	3237	100000	64897	806516
3775744	15.06.2020 12:15:47	35.80	94	114	-3.02	7.92	4.94	nano33	3108	100000	152798	12917167
3775743	15.06.2020 12:15:42	35.60	96	69	-5.88	6.79	4.75	nano33	3237	100000	64897	177
3775742	15.06.2020 12:15:38	36.31	95	84	-2.55	9.22	3.61	nano33	3237	100000	64897	12917167
3775741	15.06.2020 12:15:34	35.86	94	86	-2.55	9.22	3.53	nano33	3108	100000	152798	9332177

- Continuous monitoring can give you more insights also on the asymptomatic conditions based on the evolution of the SO2 values.

MultiSenseCrown (5/5)
Final Thoughts

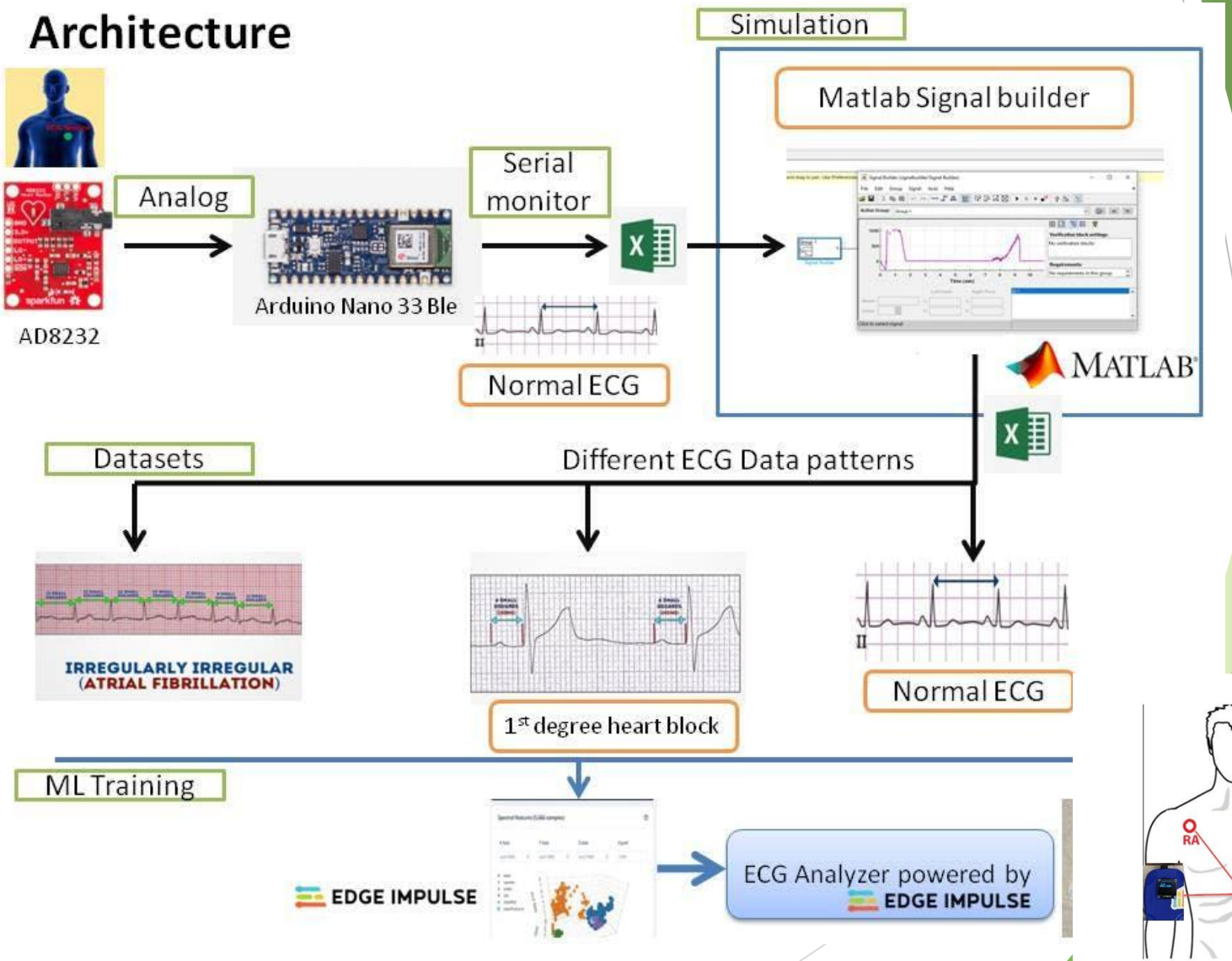
- **MultiSenseCrown goal** is to flatten the curve by breaking the disease distribution network effect while staying out of risk for patients and medical personnel.
- **Freeing hospital resources**: during pandemic there is a lack of beds and gear, if not severe conditions patients can be monitored remotely from home.
- **Direct contact people/medical**: shortage of protective gear makes physical contact highly risk for medical stuff and further virus spread
- **More patients can be monitored in less time**: having a remote monitoring system vital signs can be easily measured and instantly notified on anomalies.

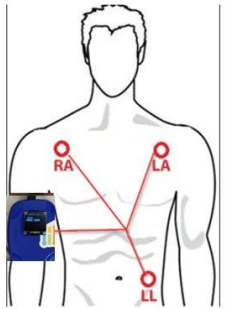
ECG Analyzer Powered by Edge Impulse (1/2)

- A **TinyML** based Medical device powered by **Edge Impulse** to predict Atrial fibrillation, AV Block 1 and Normal ECG with >90%.

- https://create.arduino.cc/projecthub/manivannan/ecg-analyzer-powered-by-edge-impulse-24a6c2

ECG Analyzer Powered by Edge Impulse (2/2)





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
  - pinMode(13, OUTPUT); // sets the digital pin 13 as output
- 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

DELAY (1/3)

- Delays are essential in **embedded systems**, unlike high-performance systems (HPC) where we want the program to **execute as fast as possible**
- Delays are used to **synchronize events**, or **read inputs** with a **specific sampling frequency**

DELAY (2/3)

- Arduino example:

```
delay(int milliseconds)
//creates a delay in ms
delayMicroseconds(int microseconds)
//creates a delay in µs

delay(1000); //one second delay
delayMicroseconds(10); //10 µs delay
```

DELAY (3/3)

- Okay, so how do we **build a delay function**?
- Our reference is the **system clock frequency**
- We use a **register or a timer** to measure ticks
- **Each tick is 1/frequency**
- Example: Assuming a processor, **an increment and a jump instruction is 1-cycle each** and a **10 MHz system clock**, build a 1-sec delay:
- T = 1/10 MHz = 100 ns
- 1 s/100 ns = 10,000,000

```
    int i=5000000; //2 cycles per iteration
BACK: i--;
    if (i!=0) goto BACK;
```

Infinite Loop (1/2)

- Embedded Systems are **mostly single-functioned**
- Their core **application never terminates**
- Infinite loops are **not forbidden** as long as **they are done correctly**

Infinite Loop (2/2)

```
void main()
{
  light enum {RED, ORANGE, GREEN};
   loop: light = RED;  //no exit from loop!
    delay(20000); //20-sec red
    light = ORANGE;
    delay(2000);  //2-sed orange
    light = GREEN;
    delay(20000); //20-sec green
    goto loop; //just repeat sequence
}
```

Example: Arduino

- Because **goto** is considered a **bad programming practice** and in order to avoid the label, **Arduino provides an infinite loop function**

```
light enum {RED, ORANGE, GREEN};
void loop(){ //the whole function repeats
   light = RED;  //no need for label!
   delay(20000); //20-sec red
   light = ORANGE;
   delay(2000);  //2-sed orange
   light = GREEN;
   delay(20000); //20-sec green
}
```

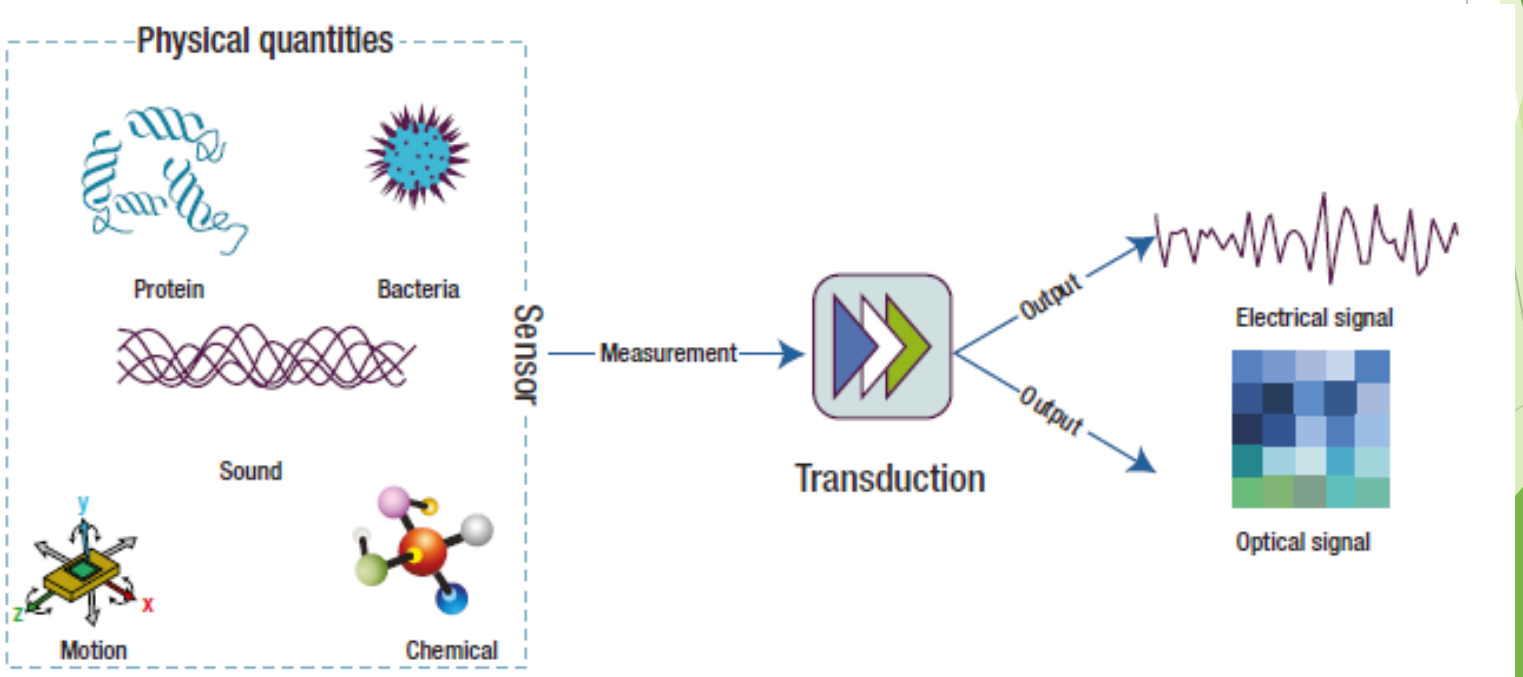
General Input/Output

- Embedded processors **receive input from their environment** (**sensors**) and **produce output that change that environment** (**actuators**) or **give information** (**indicators**)
- These require **reading and writing to single or multiple bit I/O ports**

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
  - Gyroscopes
- Optical
  - Photodetectors
  - Infrared
- Semiconductor
  - Gas
  - Temperature
  - Magnetic

Example: Simple temperature sensor

- A **RTD** (Resistance Temperature Detector) is a ***thermoresistive temperature sensor***. *It is a* ***metal element*** *(in a ceramic tube)* ***whose resistance*** **typically increases with temperature, according to a known function.**
- A linear approximation is given by $R = R_0(1 + \alpha T)$
- Where **a** is the temperature **coefficient**, **T** is the **temperature** in Kelvin and **R0** the **resistance in a known temperature**

Sensor Characteristics (1/4)

- Range
  - Full Scale Range (e.g. temperature)
  - Operating **Voltage** Range
- Accuracy
- Transfer Function
  - **S=F(x)**, where x is the measurand and S the electrical signal (commonly Voltage)
- Sensitivity
  - The **change in input required to generate a unit change** in output

Sensor Characteristics (2/4)

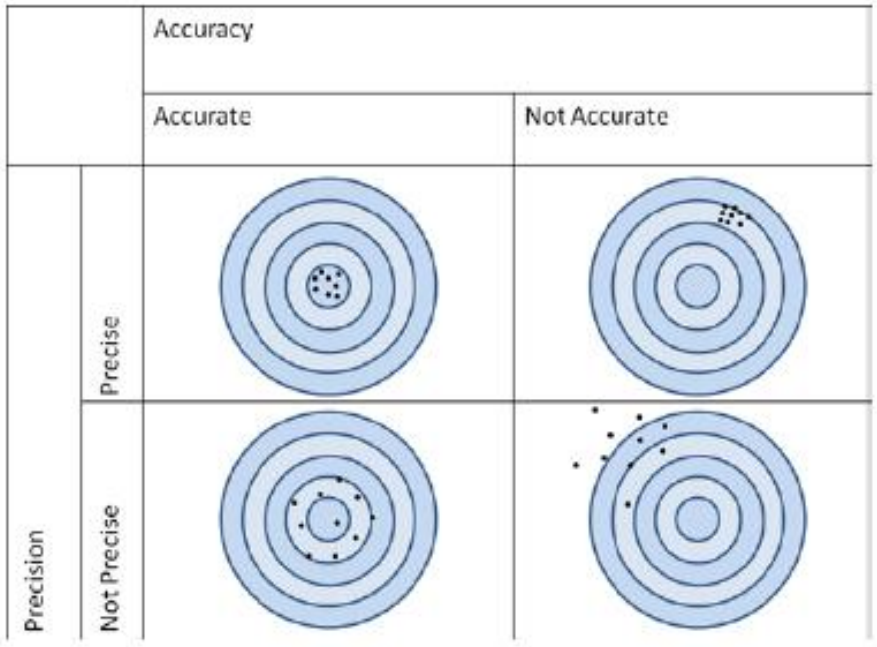
- Accuracy

$$Percentage\ Relative\ Error = \frac{(Measured\ Value - True\ Value)}{(True\ Value)} \times 100$$

  - describes the **difference between the measurement and the part's actual value**
- Precision

$$Percentage\ Standard\ Deviation = \frac{(Standard\ Deviation)}{(Mean)} \times 100$$

  - **refers to how close measurements are to each other**

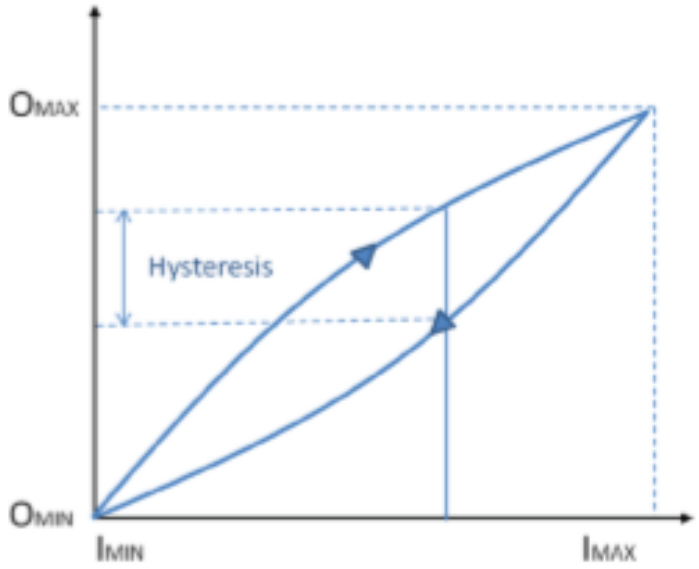


Sensor Characteristics (3/4)

- Error (Accuracy): the **difference between the measured value and true value**
- **Systematic errors** **are reproducible inaccuracies** that **can be corrected** with compensation **methods**
- **Random error**
  - Noise

Sensor Characteristics (4/4)

- **Hysteresis: the difference in output between the rising and falling output** values for a given input
- A well-known device that exhibits hysteresis is a **thermostat**
  - e.g. a thermostat is **built with hysteresis**: it will switch on heating at (say) 69°F, but switch off heating at 71°F
  - avoids the **continual** switches



Example: Smoke sensor (1/2)

- An MQ-2 **smoke sensor reports smoke by the voltage level it puts out.**
- The **more smoke there is**, the **higher the voltage**
- **built-in potentiometer** for **adjusting sensitivity**
- Three pins:
  - Vdd input
  - Ground input
  - Analog output

1 = Output
2 = Vcc (positive voltage)
3 = Gnd

Smoke sensor (2/2)

```
const int smokePin = 54; //sensor input
void setup() {
  pinMode(smokePin, INPUT);
  Serial.begin(9600);
}
void loop() {
int smoke_level = analogRead(smokePin); //read sensor
Serial.println(smoke_level);
  if(smoke_level > 120) {      //calibrate accordingly
    Serial.println("Smoke detected");
  }
  delay(100); // ms
}
```

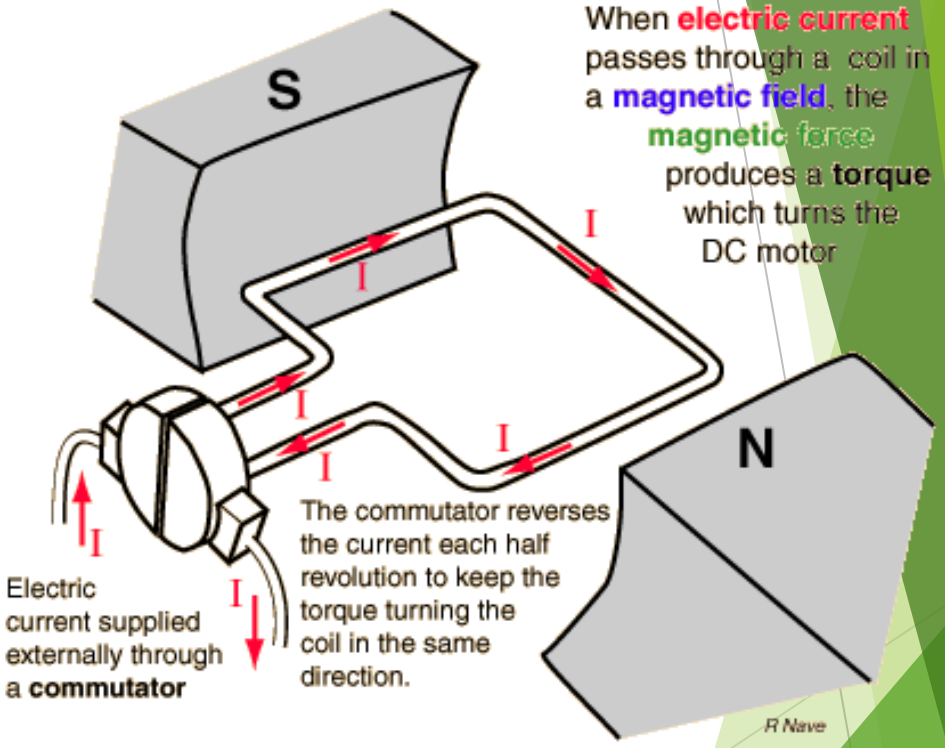
Introduction to Actuators

Actuators

- Device that **turns energy** (typically electrical) **to motion**
- Features
  - Force
  - Speed
  - Torque
  - Power
  - Efficiency

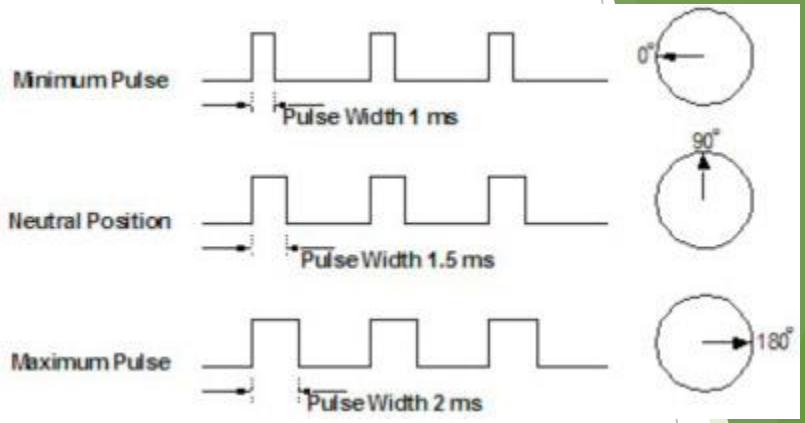
DC motor

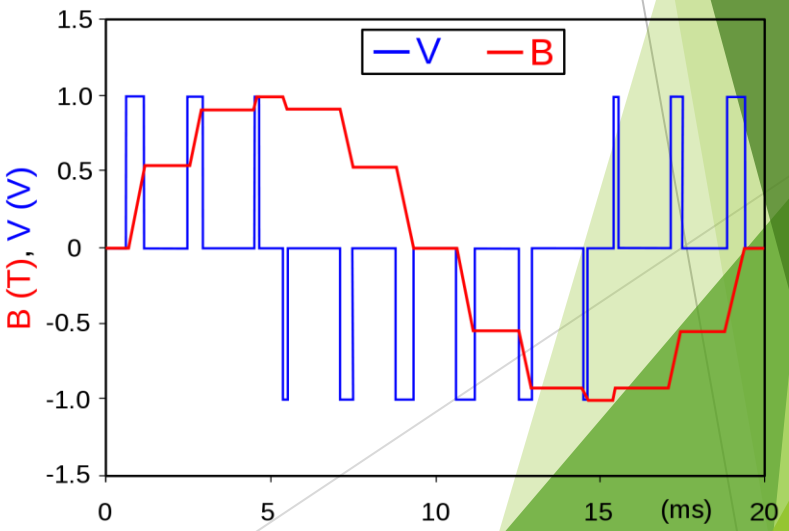
- **Force is produced** due to the **electric current in a wire inside a magnetic field.**
- Proportional to the current, therefore can be **controlled by potentiometer**
- **Hard to control precisely**



Servo motors

- **A DC motor with a control circuit**
- Servo motors are controlled by **PWM** through the control pin
- **PWM** is a method of reducing the average power delivered by an electrical signal, by effectively chopping it up into discrete parts



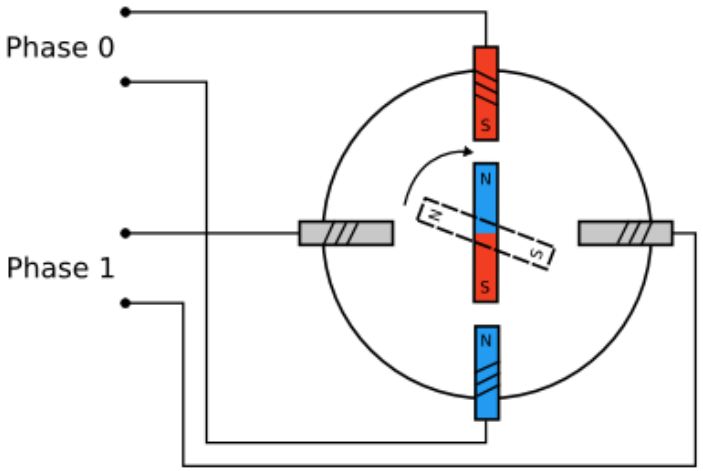


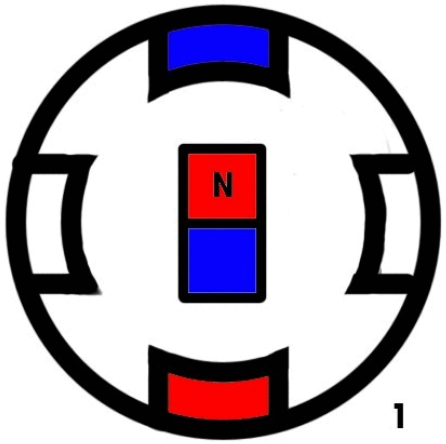
Servo motor control

```
#include <Servo.h>
    Servo myservo;  // create servo object to control a servo
    void setup() {
       myservo.attach(9);  // pin that the servo is attached to
    }
    void loop() {
       myservo.write(90);       // set position
       delay(15);                 // waits for the servo to get there

      myservo.write(180);

      delay(15);                 // waits for the servo to get there
    }
```

Stepper Motors

- motor **divides a full rotation into a number of equal steps**
- motor **controlled by a series of electromagnetic coils**
- The **center** shaft has a **series of magnets mounted on it**
- the **coils are alternately given current or not, creating magnetic fields which repulse (attract the magnets)** on the shaft, **causing the motor to rotate**
- **allows for very precise control** of the motor. It can be turned in very **accurate steps of set degree increments**





Stepper motor control

```
#include <Stepper.h> //the control sequence is in this library

const int stepsPerRevolution = 200;  // motor-dependent

Stepper myStepper(stepsPerRevolution, 8, 9, 10, 11); //pins used

void setup() {
   // set the speed at 60 rpm
   myStepper.setSpeed(60);  //actually sets the delay between steps
}

void loop() {
   // step one revolution  in one direction
   myStepper.step(stepsPerRevolution);
   delay(500);

}
```

High-Level Language Extensions for Embedded Computing

Language Extensions

- Commands in high-level languages that **are not part of the language standard** (for example ANSI C)
- Useful for **accessing low-level** (assembly) **functionality** from **a high-level language** or **simplifying desirable embedded system functionality** that is considered bad programming practice (such as infinite loops)
- Essentially **functions the user calls**
- Generally machine and **compiler-dependent**, it is possible to write an equivalent function for another machine

Examples (Arduino)

```
void loop (){
digitalWrite(77,HIGH);
delay(500);
digitalWrite(77, LOW);
}
```

This creates an infinite loop without the programmer using a goto (which is bad programming practice)

These turn pin 77 on and off (assuming there is something like a LED there without using logic instructions and I/O ports

This creates a half-second (500 ms) delay without directly accessing the timers

Using LEDs

```
void setup()
{
    pinMode(77, OUTPUT);    //configure pin 77 as output
}
// blink an LED once
void blink1()
{
    digitalWrite(77,HIGH); // turn the LED on
    delay(500); // wait 500 milliseconds
    digitalWrite(77,LOW); // turn the LED off
    delay(500); // wait 500 milliseconds
}
```

Creating infinite loops

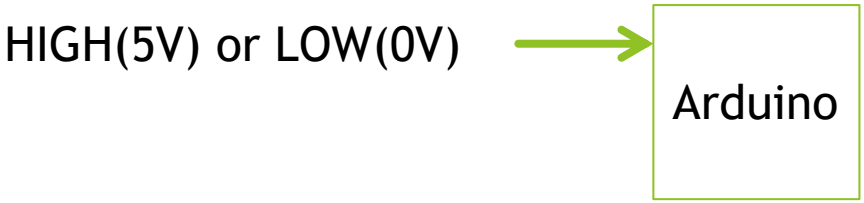
```
void loop()  //blink a LED repeatedly
{
  digitalWrite(77,HIGH); // turn the LED on
  delay(500); // wait 500 milliseconds
  digitalWrite(77,LOW); // turn the LED off
  delay(500); // wait 500 milliseconds
}
```

Using switches and buttons

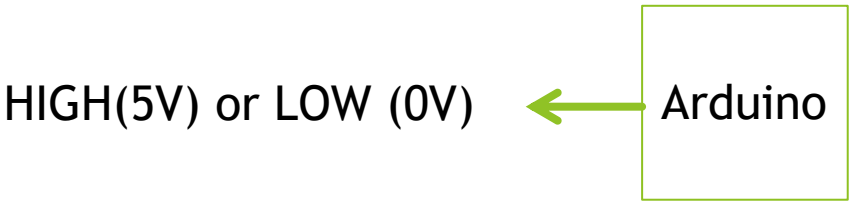
```
const int inputPin = 2; // choose the input pin
void setup() {
  pinMode(inputPin, INPUT); // declare
                            pushbutton as input
}
void loop(){
  int val = digitalRead(inputPin); // read input value
}
```

Meaning of INPUT, OUTPUT, INPUT_PULLUP

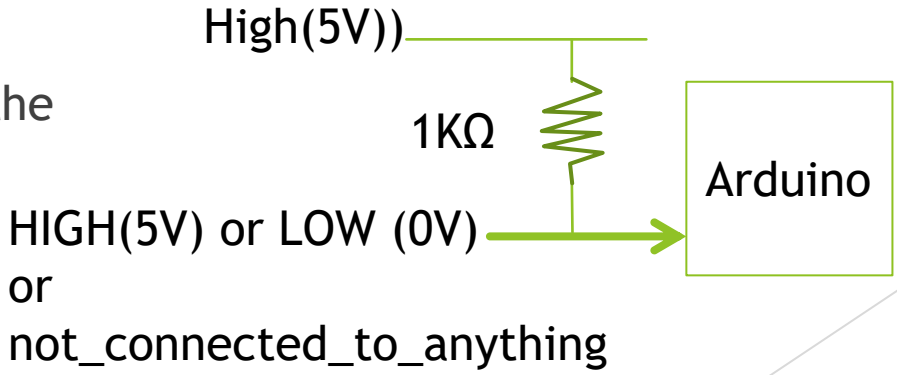
- INPUT:



- OUTPUT:



- INPUT_PULLUP: When the pin is not connect to anything, it is HIGH



Basic Control Structure – IF

```
IF(condition1){
   // do stuff if condition1 is true
}ELSE IF (condition2){
   // do stuff only if condition1 is false
   // and conition2 is true
}ELSE{
   // do stuff when both condition1 and
   // condition2 are false
}
```

Basic Control Structure – FOR

```
FOR(initialization; condition; increment){
   // statement(s);
}
```

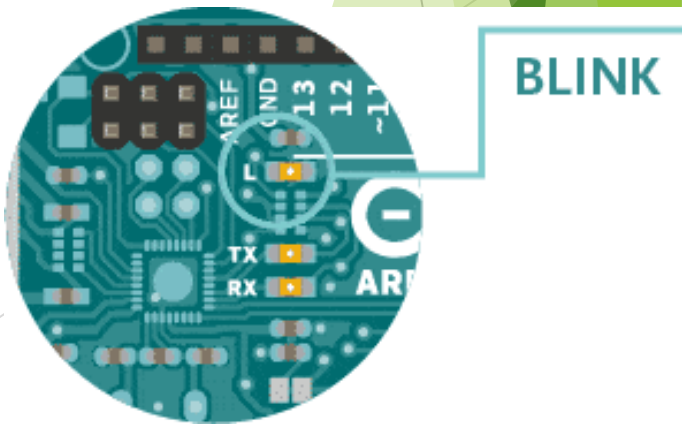
Basic Control Structure - SWITCH-CASE

```
switch (var) {
   case label1:
     // statements when var=label1
     break;
   case label2:
     // statements when var=label2
     break;
   default:
     // statements
}
```

Online IDE Example: Make your board blink from the browser

https://www.arduino.cc/en/Tutorial/BuiltInExamples

- Connect your Arduino board to your computer
  - Boards and serial ports are auto-discovered and selectable in a single dropdown
- Examples → Basic → Blink
- To upload it to your board, press the ‘Upload’
- You should then see on your board the yellow LED with an L next to it start blinking



Online IDE Example: Make your board blink from the browser

14 digital input/output (I/O) pins
13,12,… ……..2,1,0

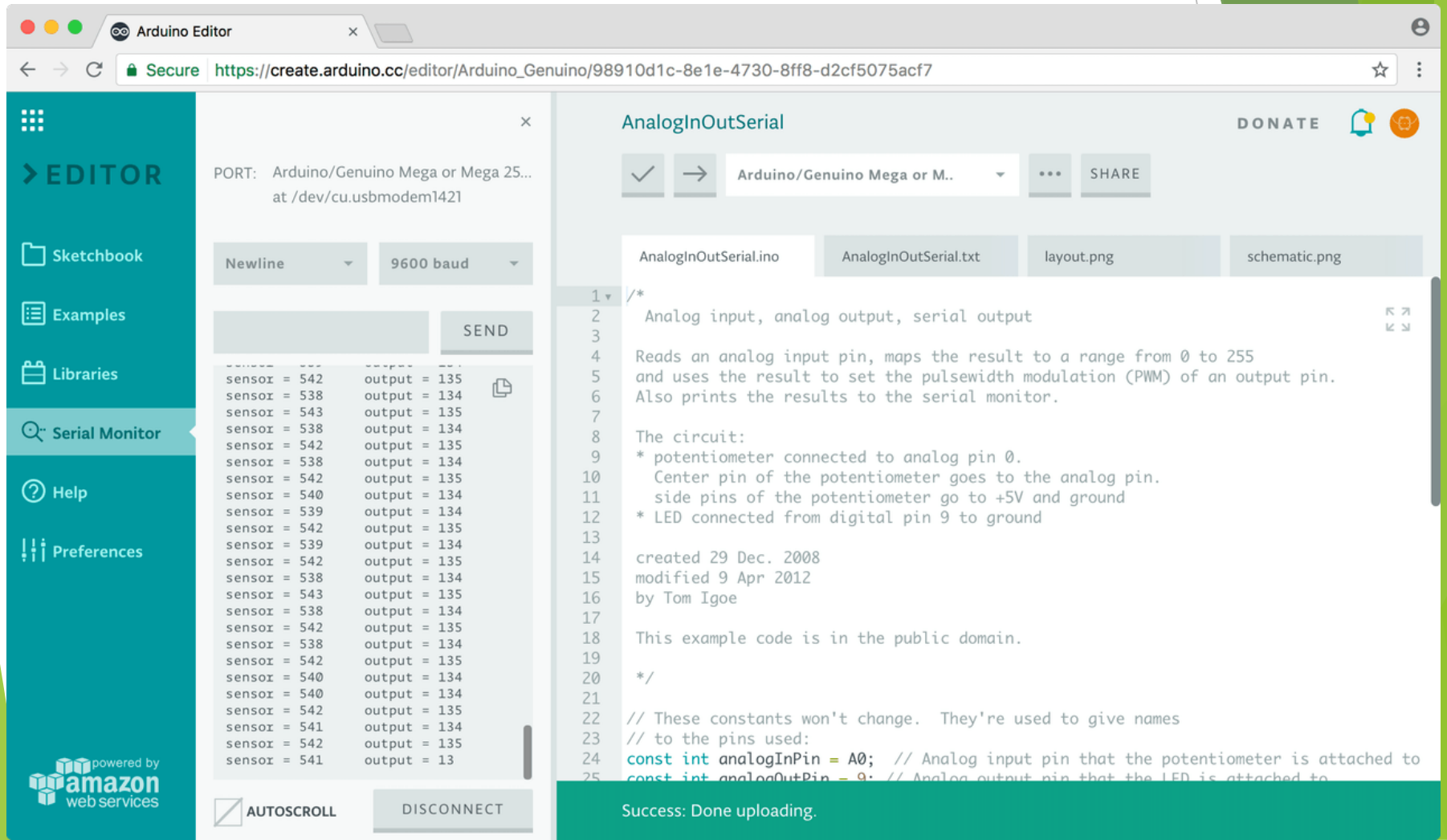
Success Connection: LED will be ON

Connect to PC through USB Cable

6 Analog inputs (or Digital I/O)
A5←A0
Can be used as digital I/O too

Online IDE Example: Serial Monitor

- If you want to **print something** and get familiar with the **Serial Monitor**
  - you can try Examples → 03.Analog → **AnalogInOutSerial**



Reading analog inputs and scaling

```
const int potPin = 0; // select the input pin for the potentiometer
void loop() {
  int val; // The value coming from the sensor
  int percent; // The mapped value
  val = analogRead(potPin); // read the voltage on the pot
                            (val ranges from 0 to 1023)
  percent = map(val,0,1023,0,100); // percent will range
                                   from 0 to 100.
```

Syntax

```
map(value, fromLow, fromHigh, toLow, toHigh)
```

https://docs.arduino.cc/built-in-examples/analog/Calibration

Creating a bar graph using LEDs (1/2)

```
const int NoLEDs = 8;
const int ledPins[] = {70, 71, 72, 73, 74, 75, 76, 77};
const int analogInPin = 0; // Analog input pin
const boolean LED_ON = HIGH;
const boolean LED_OFF = LOW;
int sensorValue = 0; // value read from the sensor
int ledLevel = 0; // sensor value converted into LED 'bars'
void setup() {
  for (int i = 0; i < NoLEDs; i++)
  {
    pinMode(ledPins[i], OUTPUT); // make all the LED pins outputs
  }
}
```

Creating a bar graph using LEDs (2/2)

```
void loop() {
sensorValue = analogRead(analogInPin); //read the analog in value

ledLevel = map(sensorValue, 0, 1023, 0, NoLEDs); //map to the
                                                    number of LEDs
for (int i = 0; i < NoLEDs; i++)
{
  if (i < ledLevel) {
    digitalWrite(ledPins[i], LED_ON); // turn on pins less than
                                         the level
  }
  else {
    digitalWrite(ledPins[i], LED_OFF); // turn off pins higher
                                          than the level
  }
}
}
```

Measuring Temperature

```
const int inPin = 0; // analog pin
void loop()
{
  int value = analogRead(inPin);
  float millivolts = (value / 1024.0)
    * 3300; //convert to 0-3.3V analog input
  float celsius = millivolts / 10; //
  sensor output is 10mV per degree
  Celsius
  delay(1000); // wait for one second
}
```

Writing data to Serial

```
void setup()
{
   Serial.begin(9600);
}
void serialtest()
{
   int i;
   for(i=0; i<10; i++)
     Serial.println(i); //human-readable ASCII text
     Serial.write(i);   //byte or series of bytes
}
```

https://www.arduino.cc/reference/en/language/functions/communication/serial/write/

https://arduino.stackexchange.com/questions/10088/what-is-the-difference-between-serial-write-and-serial-print-and-when-are-they

Reading data from Serial

```
int incomingByte = 0;   // for incoming serial data

void setup() {
        Serial.begin(9600);     // opens serial port, sets
                                   data rate to 9600 bps
}

void loop() {

        // send data only when you receive data:
        if (Serial.available() > 0) {
                // read the incoming byte:
                incomingByte = Serial.read();

                // say what you got:
                Serial.print("I received: ");
                Serial.println(incomingByte, DEC);
        }
}
```

https://www.arduino.cc/reference/en/language/functions/communication/serial/read/

Connecting LCDs

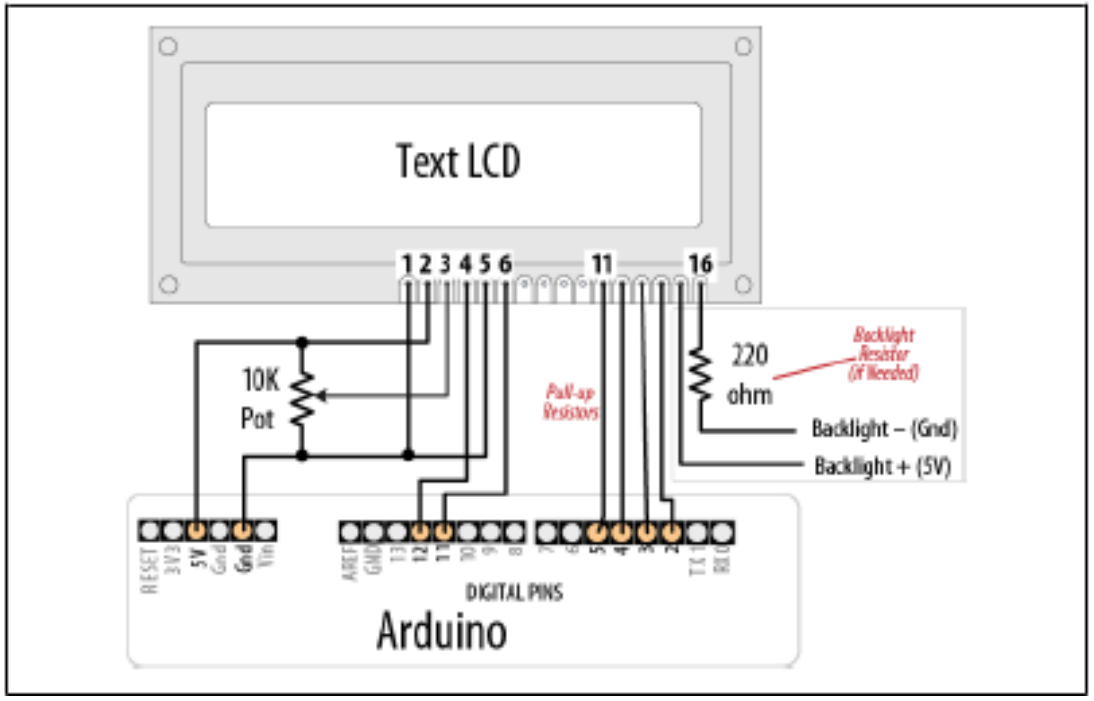


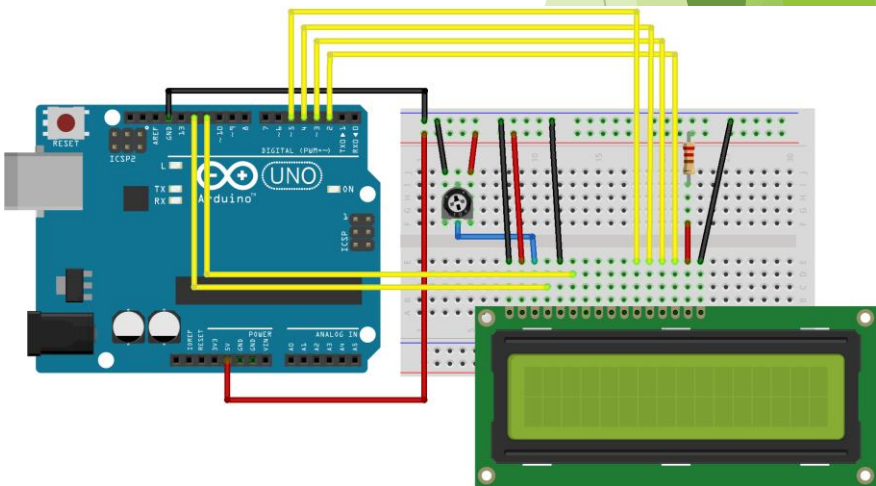
Table 11-1. LCD pin connections

LCD pin	Function	Arduino pin
1	Gnd or 0V or Vss	Gnd
2	+5V or Vdd	5V
3	Vo or contrast	
4	RS	12
5	R/W	
6	E	11
7	D0	
8	D1	
9	D2	
10	D3	
11	D4	5
12	D5	4
13	D6	3
14	D7	2
15	A or analog	
16	K or cathode	

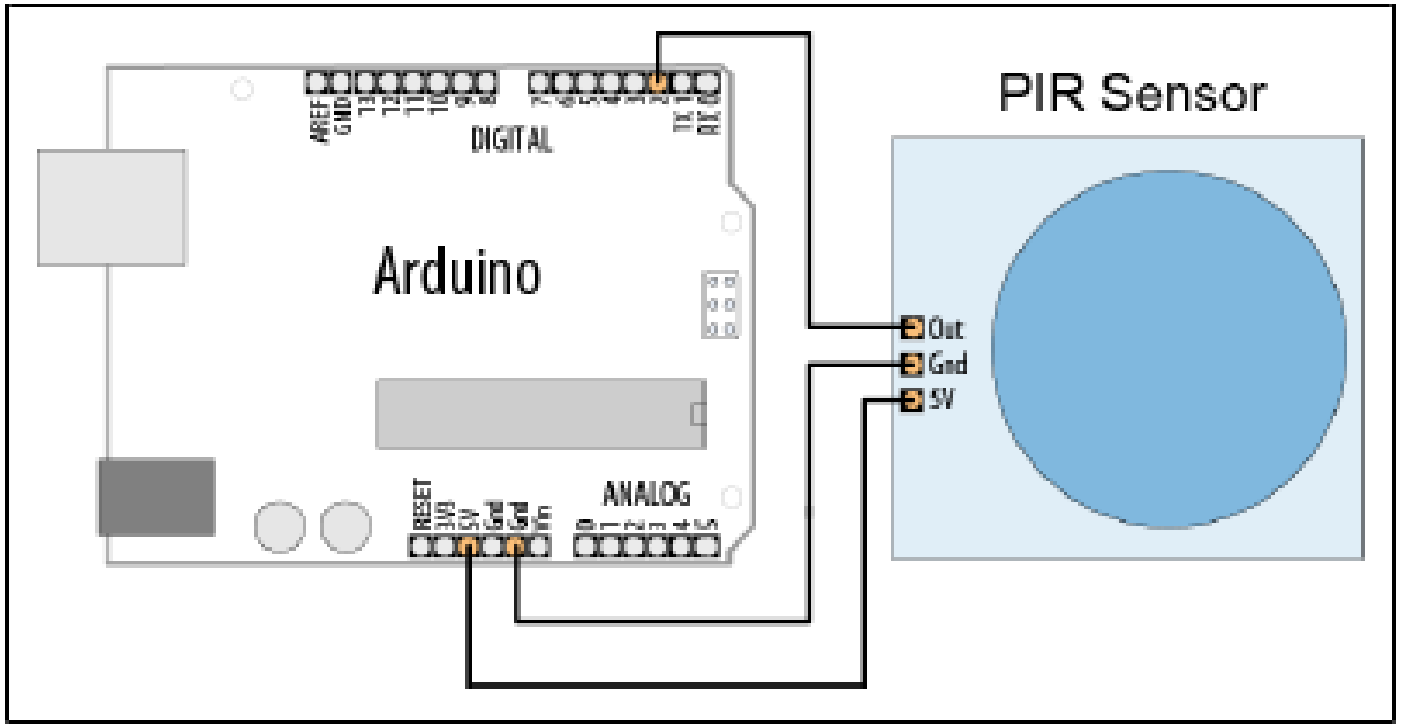
Using LCDs

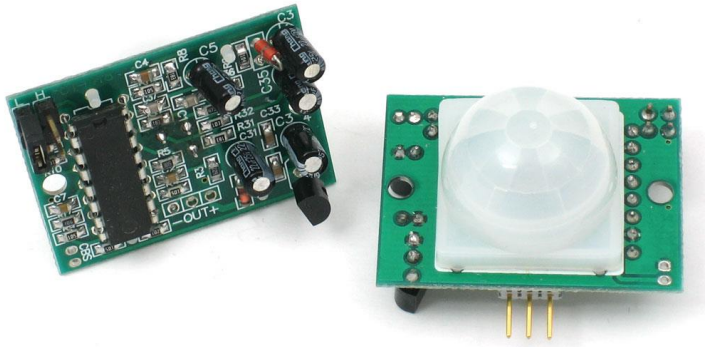
```
#include <LiquidCrystal.h> //include the library code
//constants for the number of rows and columns in the LCD
const int numRows = 2;
const int numCols = 16;

// initialize the library with the numbers of the
   interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
void setup()
{
  lcd.begin(numCols, numRows);
  lcd.print("hello, world!");
  // Print a message to the LCD
}
```

https://www.arduino.cc/en/Tutorial/LibraryExamples/LiquidCrystalBlink



Using PIR motion sensors



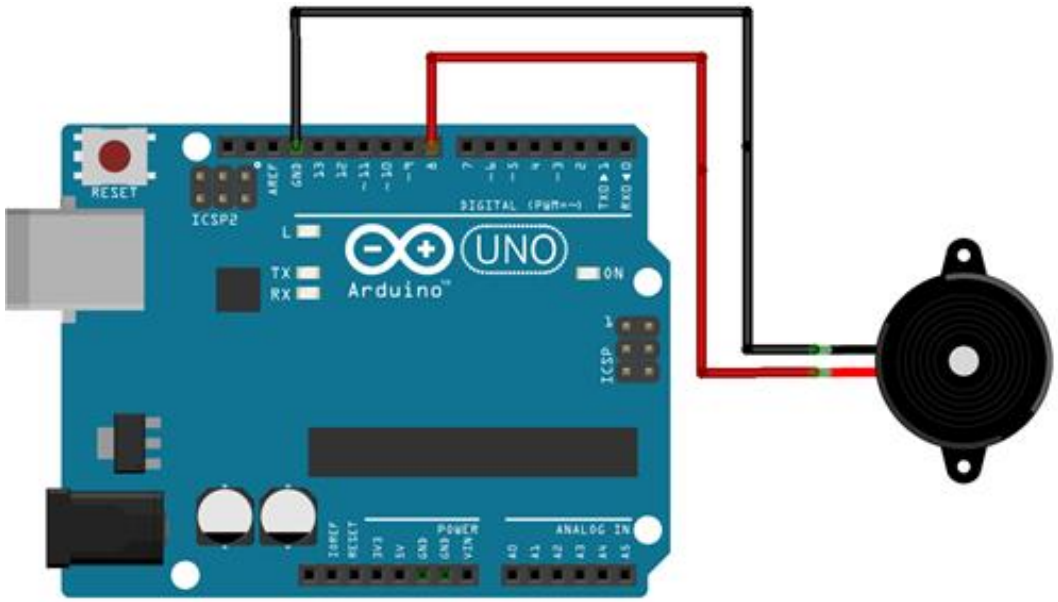


Using PIR motion sensors

```
const int ledPin = 77;  // pin for the LED
const int inputPin = 2; // input pin (for the PIR sensor)
void setup() {
  pinMode(ledPin, OUTPUT);  // declare LED as output
  pinMode(inputPin, INPUT); // declare pushbutton as input
}
void loop(){
  int val = digitalRead(inputPin); // read input value
  if (val == HIGH) // check if the input is HIGH
  {
    digitalWrite(ledPin, HIGH); //turn LED on if motion detected
    delay(500);
    digitalWrite(ledPin, LOW); // turn LED off
  }
}
```

Audio output

```
tone(speakerPin, frequency, duration);
   // play the tone
```

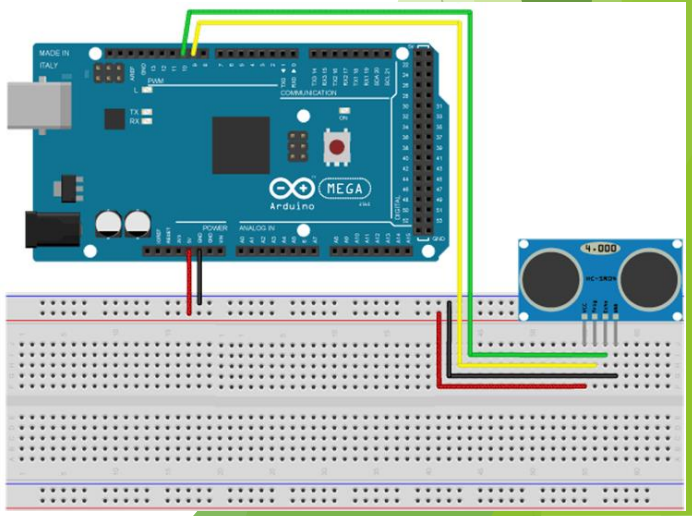


Example

- Write a program that plays an A note (440 Hz) for 1 second when a motion sensor detects motion.

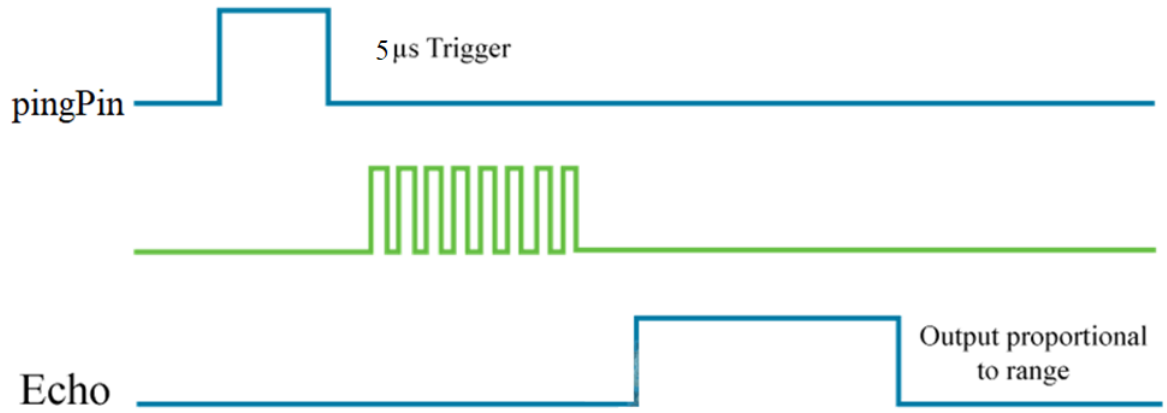
Using ultrasonic sensors (1/4)

- Ultrasonic Sensor
  - electronic device that **measures the distance** of a target object
  - **emitting ultrasonic sound waves**
  - **converts the reflected sound into an electrical signal**
- Ultrasonic sensors have two main components
  - the **transmitter** (emits the sound using piezoelectric crystals)
  - the **receiver** (which encounters the sound after it has travelled to and from the target)



Using ultrasonic sensors (2/4)

- The “ping” sound pulse is generated when the **pingPin level goes HIGH** for 5 microseconds
- The sensor will then **generate a pulse that terminates when the sound returns**
- The **width of the pulse** is **proportional to the distance** the sound traveled
- The speed of sound is **340 meters per second**, which is **29 microseconds per centimeter**. The formula for the distance of the round trip is:
  - RoundTrip = microseconds / 29



Using ultrasonic sensors (3/4)

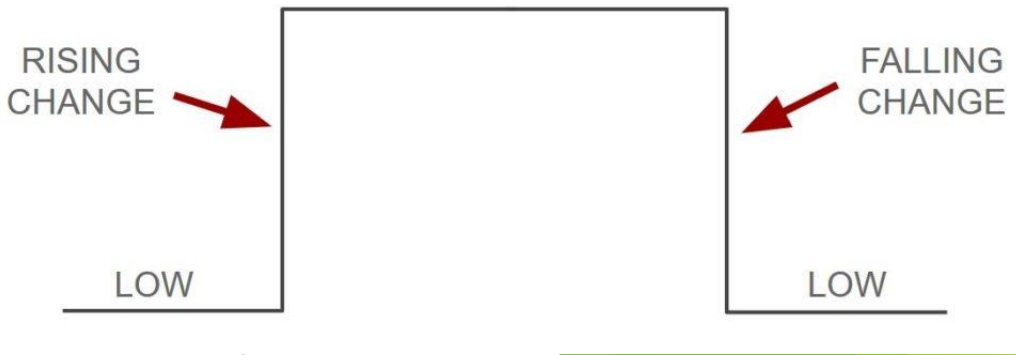
```
const int pingPin = 5;
const int ledPin = 77; // pin connected to LED

void setup()
{
  Serial.begin(9600);
  pinMode(ledPin, OUTPUT);
}

void loop()
{
  int cm = ping(pingPin) ;
  Serial.println(cm);
  digitalWrite(ledPin, HIGH);
  delay(cm * 10); // each centimeter adds 10 milliseconds delay
  digitalWrite(ledPin, LOW);
  delay(cm * 10);
}
```

Using ultrasonic sensors (4/4)

```
int ping(int pingPin)
{
  long duration, cm;
  pinMode(pingPin, OUTPUT);
  digitalWrite(pingPin, LOW);
  delayMicroseconds(2);
  digitalWrite(pingPin, HIGH);
  delayMicroseconds(5);
  digitalWrite(pingPin, LOW);
  pinMode(pingPin, INPUT);
  duration = pulseIn(pingPin, HIGH); // Read a pulse (HIGH or LOW) on a pin
  cm = microsecondsToCentimeters(duration); // convert the time into a distance
  return cm ;
}

long microsecondsToCentimeters(long microseconds)
{
  // The speed of sound is 340 m/s or 29 microseconds per centimeter.
  // The ping travels out and back, so to find the distance of the
  // object we take half of the distance travelled.
  return microseconds / 29 / 2;
}
```

Using Interrupts

- On a standard Arduino board, **two pins can be used as interrupts: pins 2 and 3**
- The interrupt is enabled through the following line:
  - attachInterrupt(interrupt, function, mode)
    - attachInterrupt(3, doEncoder, FALLING);
- Triggered:
  - **RISING**: LOW to HIGH
  - **FALLING**: HIGH to LOW
  - **CHANGE**: signal changes
    - LOW to HIGH or HIGH to LOW
  - **LOW**: signal is LOW



Blink example with Interrupt

```
int led = 77;
volatile int state = LOW;

void setup()
{
  pinMode(led, OUTPUT);
  attachInterrupt(2, blink, CHANGE);
}

void loop()
{
  digitalWrite(led, state);
}

void blink()
{
  state = !state;
}
```

Timer functions (timer.h library)

- int every(long period, callback)
  - **Run the 'callback' every 'period' milliseconds.** Returns the ID of the timer event.
- int every(long period, callback, int repeatCount)
  - **Run the 'callback' every 'period' milliseconds for a total of 'repeatCount' times.** Returns the ID of the timer event.
- int after(long duration, callback)
  - **Run the 'callback' once after 'period' milliseconds.** Returns the ID of the timer event.

Timer functions (timer.h library)

- int oscillate(int pin, long period, int startingValue)
  - **Toggle the state of the digital output 'pin' every 'period' milliseconds.** The pin's starting value is specified in 'startingValue', which should be **HIGH** or **LOW**. Returns the ID of the timer event.
- int oscillate(int pin, long period, int startingValue, int repeatCount)
  - **Toggle the state of the digital output 'pin' every 'period' milliseconds 'repeatCount' times.** The pin's starting value is specified in 'startingValue', which should be **HIGH** or **LOW**. Returns the ID of the timer event.
- Example: How to implement the blink example using oscillate?

Timer functions (timer.h library)

- int pulse(int pin, long period, int startingValue)
  - **Toggle the state of the digital output 'pin' just once after 'period' milliseconds.** The pin's starting value is specified in 'startingValue', which should be **HIGH** or **LOW**. Returns the ID of the timer event.
- int stop(int **id**)
  - **Stop the timer event running.** Returns the ID of the timer event.
- int update()
  - **Must be called from 'loop'.** This will service all the events associated with the timer.

Example (1/2)

```
#include "timer.h"
Timer t;
int ledEvent;
void setup()
{
  Serial.begin(9600);
  int tickEvent = t.every(2000, doSomething); //call every 2s
  Serial.print("2 second tick started id=");
  Serial.println(tickEvent);
  pinMode(13, OUTPUT);
  ledEvent = t.oscillate(13, 50, HIGH); //Toggle the state of 13 every 50ms
  Serial.print("LED event started id=");
  Serial.println(ledEvent);
  int afterEvent = t.after(10000, doAfter); //Run the 'doAfter' once after 10s
  Serial.print("After event started id=");
  Serial.println(afterEvent);
}
```

Example (2/2)

```
void loop()
{
  t.update();
}

void doSomething()
{
  Serial.print("2 second tick: millis()=");
  Serial.println(millis());
}

void doAfter()
{
  Serial.println("stop the led event");
  t.stop(ledEvent);
  t.oscillate(13, 500, HIGH, 5); //Toggle the state of 13 every 500ms for 5 times
}
```

Arduino with GPS (1/3)

- GPS **obtains information such as position** and **altitude** with **only** the **signal of the satellites**
  - *AT+CGPSINFO* command **brings directly latitude, logtitude, date, UTC time, altitude and speed**

- Example: AT+CGPSINFO returns the GPS info in a **string:**
  - [<latitude>],[<N/S>],[<longitude>],[<E/W>],[<date>],[<UTC_time>],[<altitude>],[<speedOG>],[<course>]

Arduino with GPS (2/3)

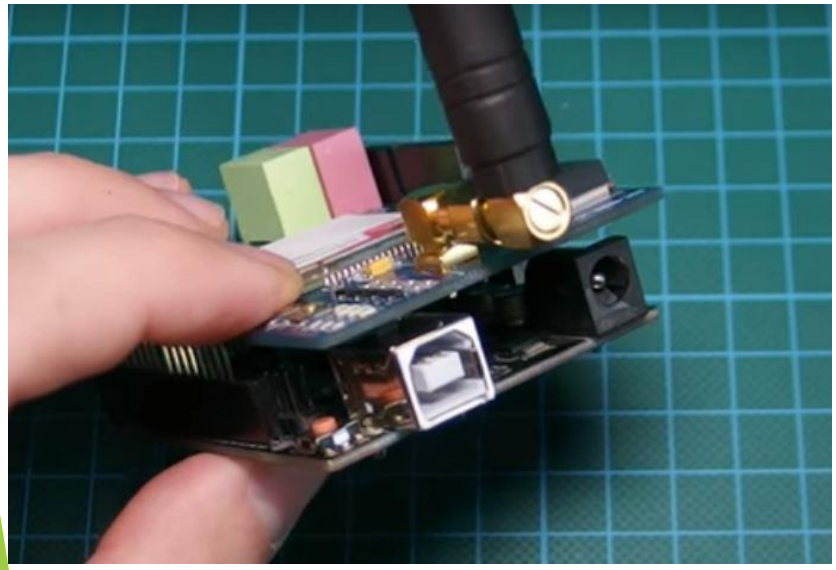
```
int8_t answer;
int onModulePin= 2;
char gps_data[100];    //will perform 100 GPS data reads
int counter;
void setup(){
    pinMode(onModulePin, OUTPUT);
    Serial.begin(115200);
    Serial.println("Starting...");
    power_on();
    delay(5000); // starts GPS session in stand alone mode
  }
}
void power_on(){    //void power_on() should be placed AFTER void loop(), used here for lecture
    digitalWrite(onModulePin,HIGH);    //GPS module requires a 3 sec pulse on onModulePin
    delay(3000);
    digitalWrite(onModulePin,LOW);
}
```

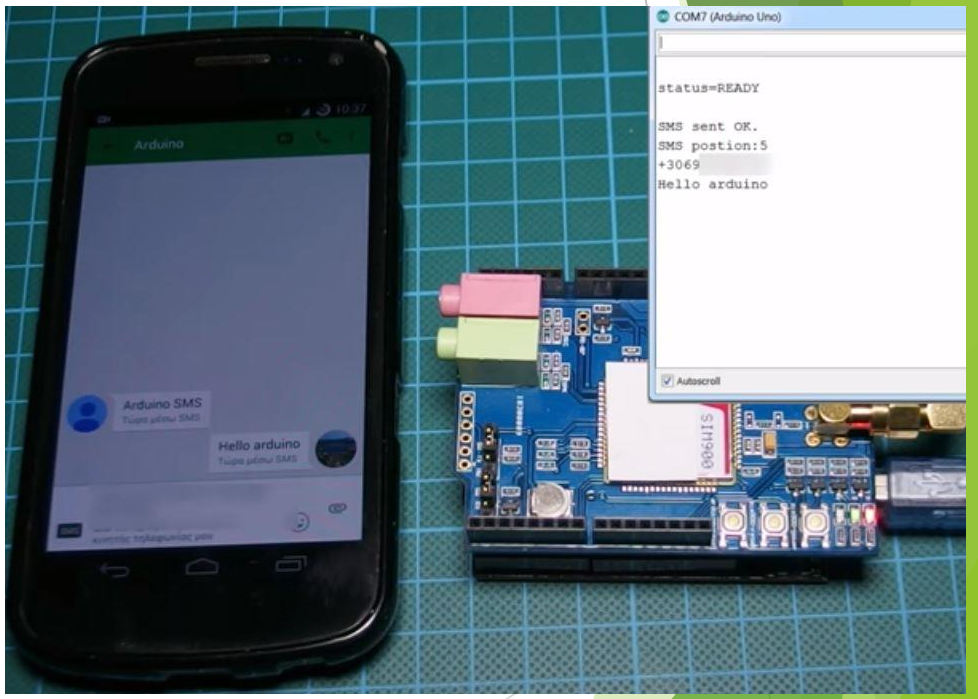
Arduino with GPS (3/3)

```
void loop(){
  answer = sendATcommand("AT+CGPSINFO","+CGPSINFO:",1000);
  // request info from GPS
  if (answer == 1) {
       counter = 0;
       do{
          gps_data[counter] = Serial.read();
          counter++;
        }
        while(Serial.available() == 0);    //reading GPS data
  }
  Serial.print("GPS data:");                //printing GPS data
  Serial.print(gps_data);
}
```

Connecting through GSM

https://www.youtube.com/watch?v=n-RkWRUw62g





Finite State Machine (FSM)

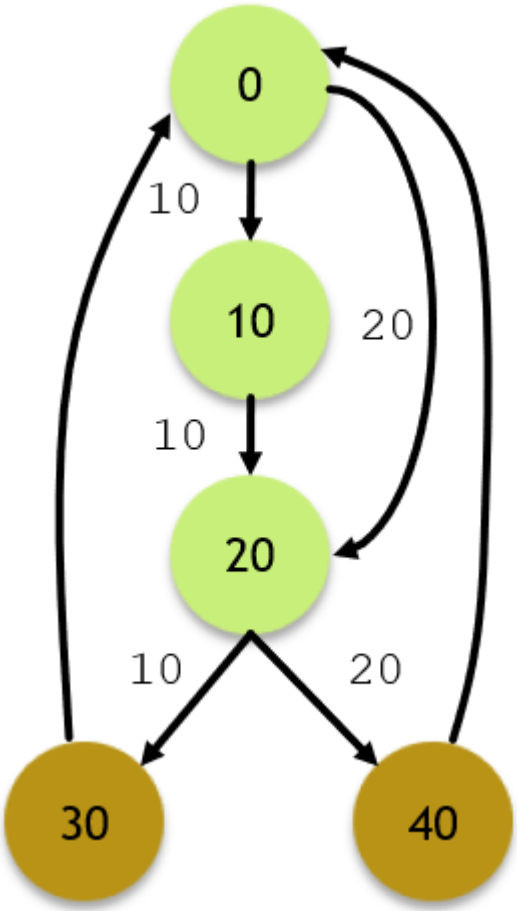
Use FSM to control a system

Logic Control

- Logic control is an essential part to develop an intelligence device

- Logic control can be developed by **Finite State Machine (FSM)**
  - Decision is based on what it has done i.e. the system has memory, the performance can be more complex

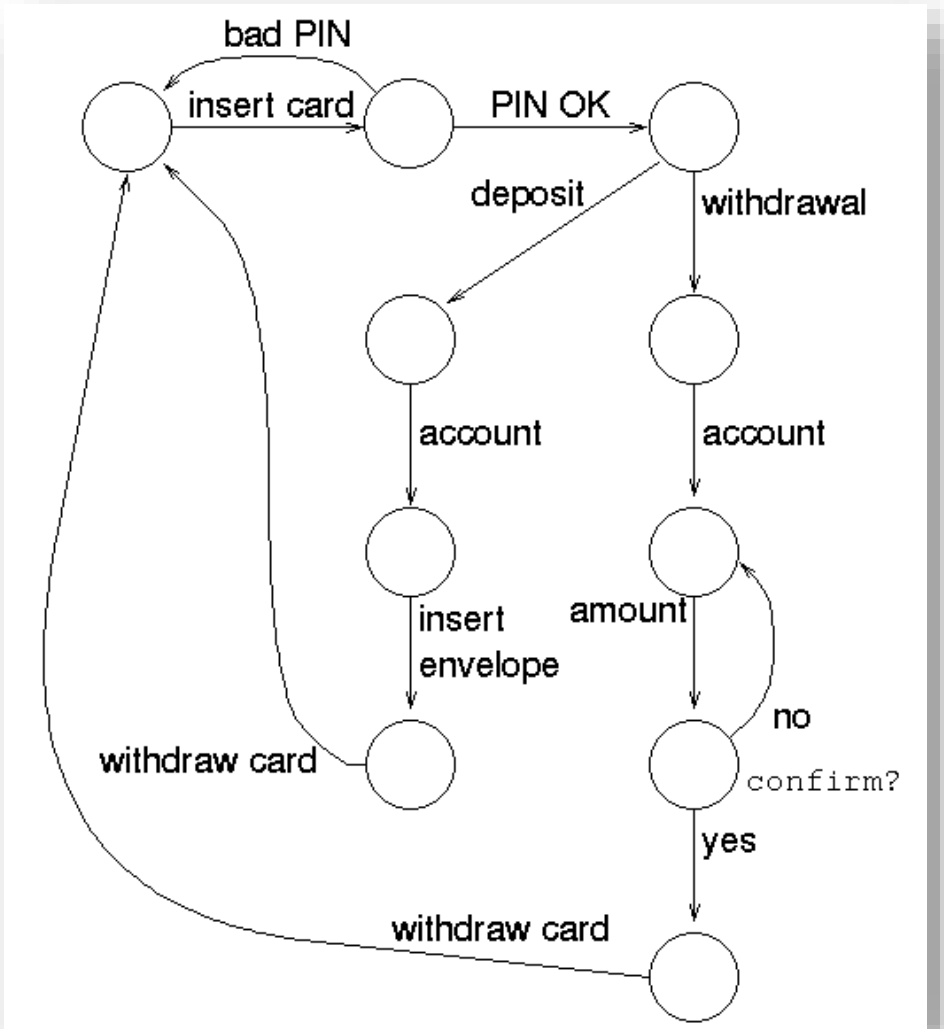
FSM Examples (1)



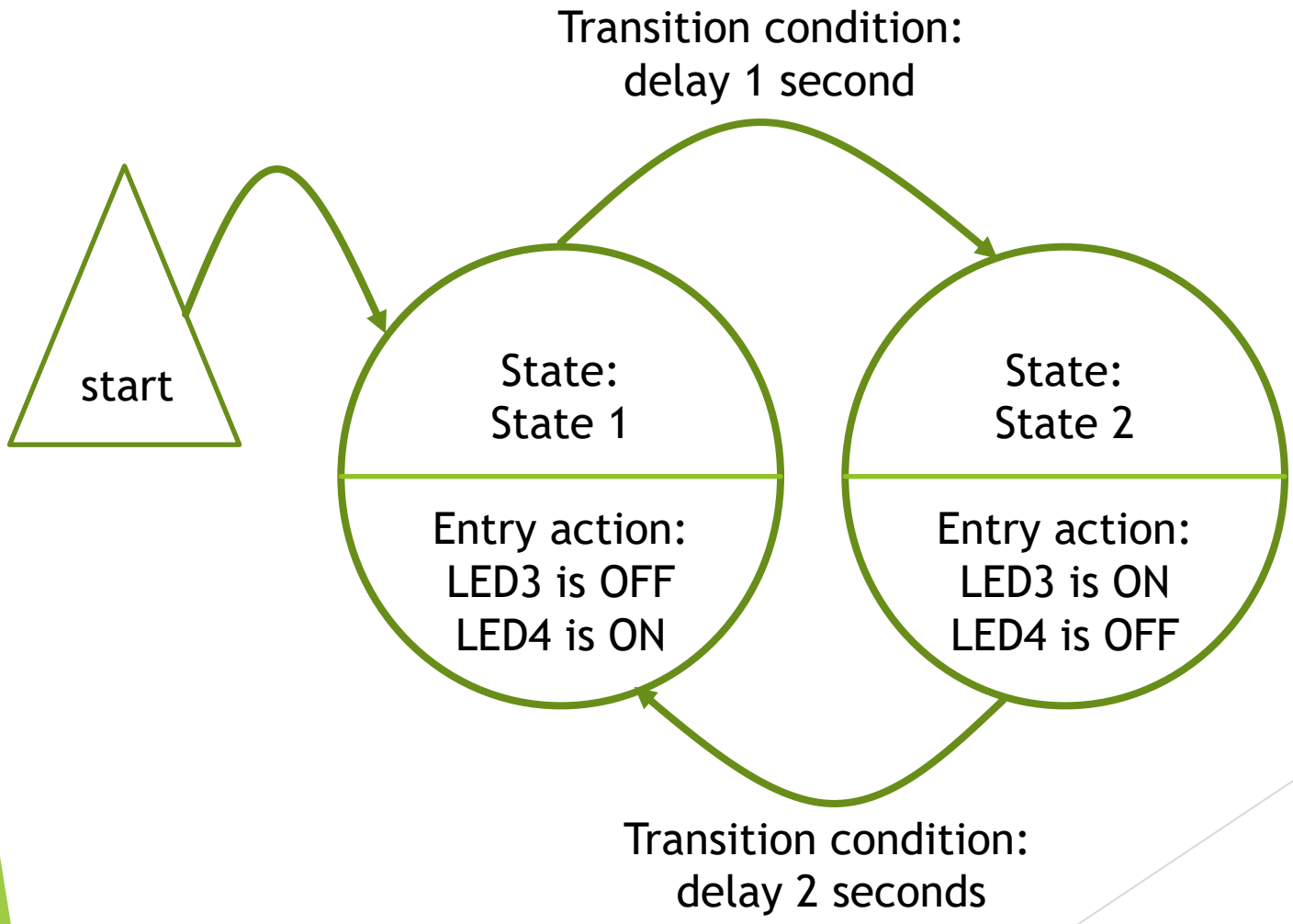
- 10c

Τιμή: 30c

FSM Examples (2)



FSM demo 1 (FSM with no input)



Code

```
#define STATE1 1
#define STATE2 2
#define STATE_END 100
//init to state1
unsigned char state=1;
void setup() {
  pinMode (3, OUTPUT);
  pinMode (4, OUTPUT);
}
```

```
void loop() {
  switch(state) {
    case STATE1:
        digitalWrite(3, HIGH); // LED OFF
        digitalWrite(4, LOW); // LED ON
        delay(1000);
        state=STATE2;
        break;
    case STATE2:
         digitalWrite(3, LOW); // LED ON
         digitalWrite(4, HIGH); // LED OFF
         delay(2000);
         state=STATE1;
         break;
     case STATE_END: // turn off LEDs, not used here
         digitalWrite(3, HIGH); // LED OFF
         digitalWrite(4, HIGH); // LED OFF
         break;
      default:
         state=STATE_END;
         break;
     }  }
```

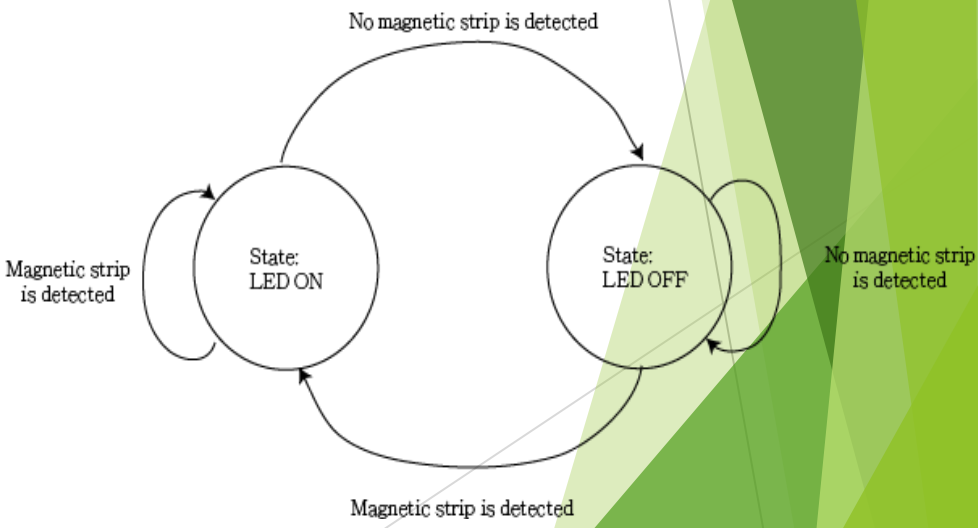
FSM demo 2
Two-State FSM with input (1)

- When a magnetic strip is detected, the LED is ON
- When there is no magnetic strip, the LED is OFF

State Transition Table

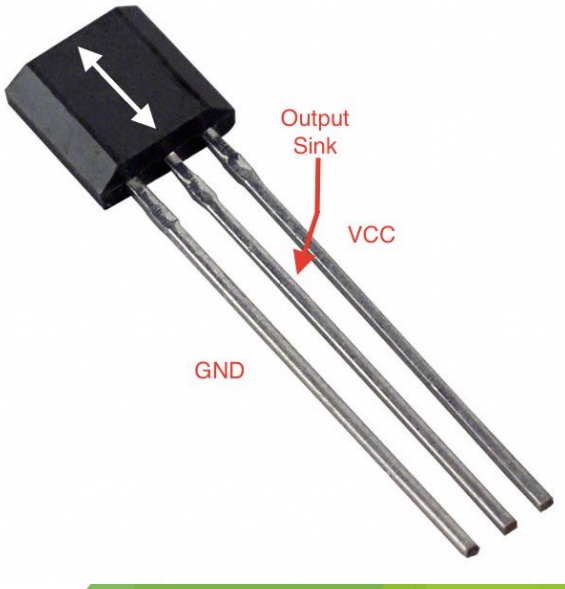
Input	Current State	Next State
Magnetic strip is detected	**ON**	ON
Magnetic strip is detected	**OFF**	ON
No magnetic strip is detected	**ON**	OFF
No magnetic strip is detected	**OFF**	OFF

State Diagram



FSM demo 2
Two-State FSM with input (2)

- Circuit
  - Connect a magnetic sensor to pin 7
- Functionality
  - When a magnet is near the magnetic switch sensor then LED5=ON,LED6=OFF
  - When a magnet is NOT near the magnetic switch sensor, then LED5=OFF,LED6=ON



Code

```
#define STATE1 1
#define STATE2 2
#define STATE_END 100
int magnetic = 7;
int ledPin_S1 = 5;
int ledPin_S2 = 6;
unsigned char state=1;
void setup() {
  pinMode (magnetic, INPUT);
  pinMode (ledPin_S1, OUTPUT);
  pinMode (ledPin_S2, OUTPUT);
}
```

```
void loop() {
  switch(state) {
    case STATE1: //a magnet is near (HIGH)
        digitalWrite(ledPin_S1, LOW); // LED ON
        digitalWrite(ledPin_S2, HIGH); // LED OFF
        if (digitalRead(magnetic) == LOW)
            state=STATE2; break;
    case STATE2: //a magnet is NOT near (LOW)
         digitalWrite(ledPin_S1, HIGH); // LED OFF
         digitalWrite(ledPin_S2, LOW); // LED ON
         if (digitalRead(magnetic) == HIGH)
            state=STATE1; break;
     case STATE_END:
         digitalWrite(ledPin_S1, HIGH); // LEDOFF
         digitalWrite(ledPin_S2, HIGH); // LEDOFF
         break;
      default:
         state=STATE_END;
         break;
}}
```

More Examples

- https://docs.arduino.cc/built-in-examples/digital/BlinkWithoutDelay
- https://docs.arduino.cc/built-in-examples/digital/Button
- https://docs.arduino.cc/built-in-examples/digital/toneMultiple
- https://docs.arduino.cc/built-in-examples/analog/AnalogInput
- https://docs.arduino.cc/built-in-examples/analog/Calibration
- https://docs.arduino.cc/built-in-examples/analog/Fading

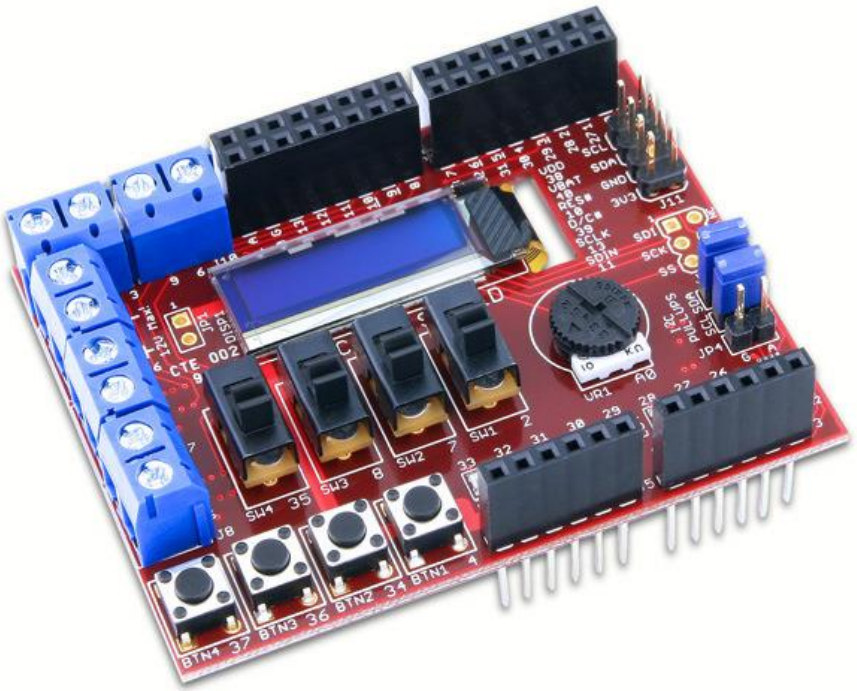
Chipkit Max32 arduino board

ChipKit MAX32

- Microcontroller: PIC32MX795F512L
- **Flash Memory: 512K**
- **RAM Memory: 128K**
- Operating Voltage: 3.3V
- **Operating Frequency: 80Mhz**
- **Typical operating current: 90mA**
- **Input Voltage** (recommended): **7V to 15V**
- Input Voltage (maximum): 20V
- **I/O Pins: 83 total**
- **Analog Inputs: 16**
- **Analog input voltage range: 0V to 3.3V**
- DC Current per pin: +/-18mA
- Advanced **peripherals**:
  - 10/100 Ethernet MAC
  - USB 2.0 Full Speed OTG controller
  - 2 CAN (Controller Area Network) controllers
- **External Interrupts**: Pin 3 (INT0), Pin 2 (INT1), Pin 7 (INT2), Pin 21 (INT3), Pin 20 (INT4)

Basic I/O Shield

- Features
  - 128x32 pixel **OLED graphic display**
  - I2C **temperature sensor**
  - 256Kbit I2C EEPROM
  - I2C daisy chain connector
  - 4 push buttons
  - 4 slide switches
  - 8 discrete LEDs
  - Analog potentiometer



Temperature Sensor

- A **digital temperature sensor is provided using a Microchip TCN75A Serial Temperature Sensor**. The temperature sensor is an I2C device, and **is located in the lower right corner of the board**.
- The TCN75A is rated for an **accuracy of +/-1°C** and has selectable **resolution from 0.5°C down to 0.0625°C**.
- **Digilent provides a library for accessing the temperature sensor**. This library is **available on the Digilent web site** and in the third party library repository on **github**.
- Using the temperature sensor with the MAX32 board requires **manually connecting** to **the basic I/O shield**

Configuring Temperature sensor

- **void config(uint8_t configuration)**
- Parameters
- configuration - Value to be written to config register
- This function writes the configuration register with the given value. There are a number of defined
- values as described below that can be or'd together to set multiple parameters. For example if one
- wished to put the device in one shot mode and use 12 bit resolution the following call could be made.
- Config(ONESHOT | RES12)
- IOSHIELDTEMP_ONESHOT 0x80 //One Shot mode
- IOSHIELDTEMP_RES9 0x00 //**9-bit resolution (0.5°C)**
- IOSHIELDTEMP_RES10 0x20 //10-bit resolution
- IOSHIELDTEMP_RES11 0x40 //11-bit resolution
- IOSHIELDTEMP_RES12 0x60 //**12-bit resolution (0.0625°C)**
- IOSHIELDTEMP_FAULT1 0x00 //1 fault queue bits
- IOSHIELDTEMP_FAULT2 0x08 //2 fault queue bits
- IOSHIELDTEMP_FAULT4 0x10 //4 fault queue bits
- IOSHIELDTEMP_FAULT6 0x18 //6 fault queue bits
- IOSHIELDTEMP_ALERTLOW 0x00 //Alert bit active-low
- IOSHIELDTEMP_ALERTHIGH 0x04 //Alert bit active-high
- IOSHIELDTEMP_CMPMODE 0x00 ///comparator mode.
- IOSHIELDTEMP_INTMODE 0x02 //interrupt mode
- IOSHIELDTEMP_STARTUP 0x00 //Shutdown disabled
- IOSHIELDTEMP_SHUTDOWN 0x01 //Shutdown enabled
- IOSHEIDLTEMP_CONF_DEFAULT //Power up initial configuration

Reading Temperature sensor

- **float getTemp()**
- Retrieves the **current temp from the** temp **sensor** and **converts the returned value** into a signed **floating point value.**

Example

```
void setup(){
IOShieldTemp.config(IOSHIELDTEMP_RES11 |
   IOSHIELDTEMP_ONESHOT);
}

void loop()
{
   //Get Temperature in Celsius
   float temp;
   temp = IOShieldTemp.getTemp();
}
```

Potentiometer

- A **potentiometer is provided on the board** to be used as an **analog signal source** or **analog control input**.
- The wiper of the pot is connected **to analog input A0**.
- The pot is read using the **analogRead function**.

OLED Display

- 128x32 pixels
- Each individual pixel can only be **on** (illuminated) or **off** (not illuminated)
- The display is a **serial device** that is accessed **via an SPI** (Serial Peripheral Interface) interface
- **write-only device**

Example

```
void setup()
{
    IOShieldOled.begin();
}

void loop()
{
    char toprint;

    IOShieldOled.clearBuffer();
    IOShieldOled.setCursor(0, 0);

    toprint = ‘A’;

    IOShield0led.putChar(toprint);
  }
```