

ME-410: Introduction to Arduino and its applications

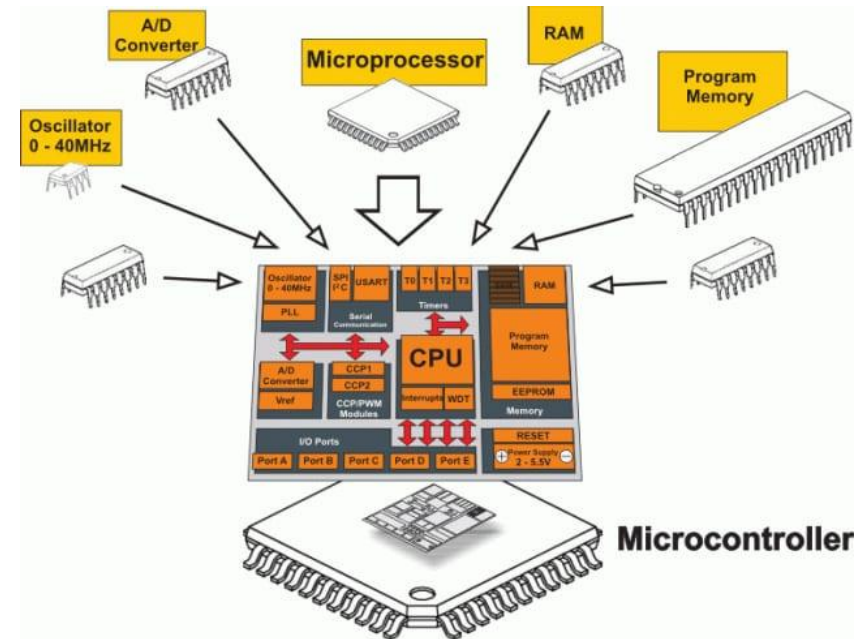
Ziqiao Wang, Prof. Jamie Paik
Reconfigurable Robotics Laboratory
EPFL, Switzerland

What is Microcontroller?

- **Definition:** A compact integrated circuit designed to govern a specific operation in an embedded system.
- **Components:**
 - Processor
 - memory
 - input/output pins
 - peripheral

Why we need microcontroller?

- Size and portability
- Power consumption
- Real-time Operation



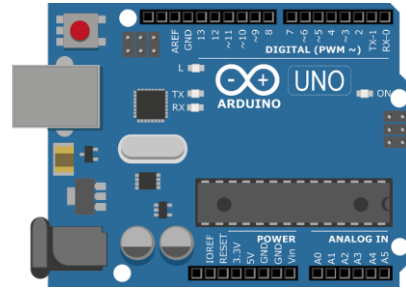
Source: Max Embedded

What is Arduino?

Open-source hardware and software company that designs and manufactures **single-board microcontrollers**

Code and designs files are accessible by anyone!

- No “black box” = more control
- Completely free
- Don't rely on a company
- Active and large community



Provides all of the circuitry necessary for a useful control task: a microprocessor, I/O circuits, a clock generator, RAM, stored program memory and any necessary support ICs.

Arduino in ME-410

Inputs

User



Sensors



Analog



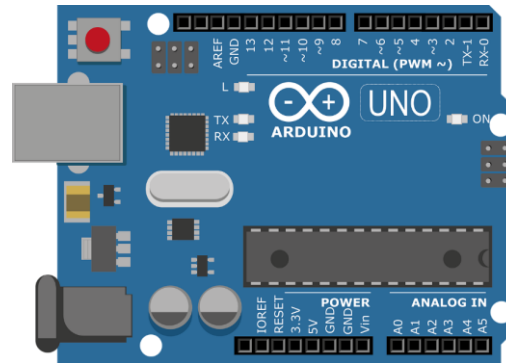
Digital



Signals

Programming

Reading



Outputs

Actuators



Indicators



Analog



Digital



Signals

Arduino can interact with electronic components to control your robot

Choose your board

Features	Specs	Integration	Sensors
• ADCs • DACs • GPIO • Wifi • Bluetooth • ...	• Clock speed • Memory • Interrupts • Power consumption • ...	• Size • Weight • Input voltage • Output voltage • Output current • ...	• Gyro • Accel • Magnetometers • Gesture • Humidity • Sound • ...

[Official arduino board](#)

Variety of Features,
Specs and Integrations
capabilities

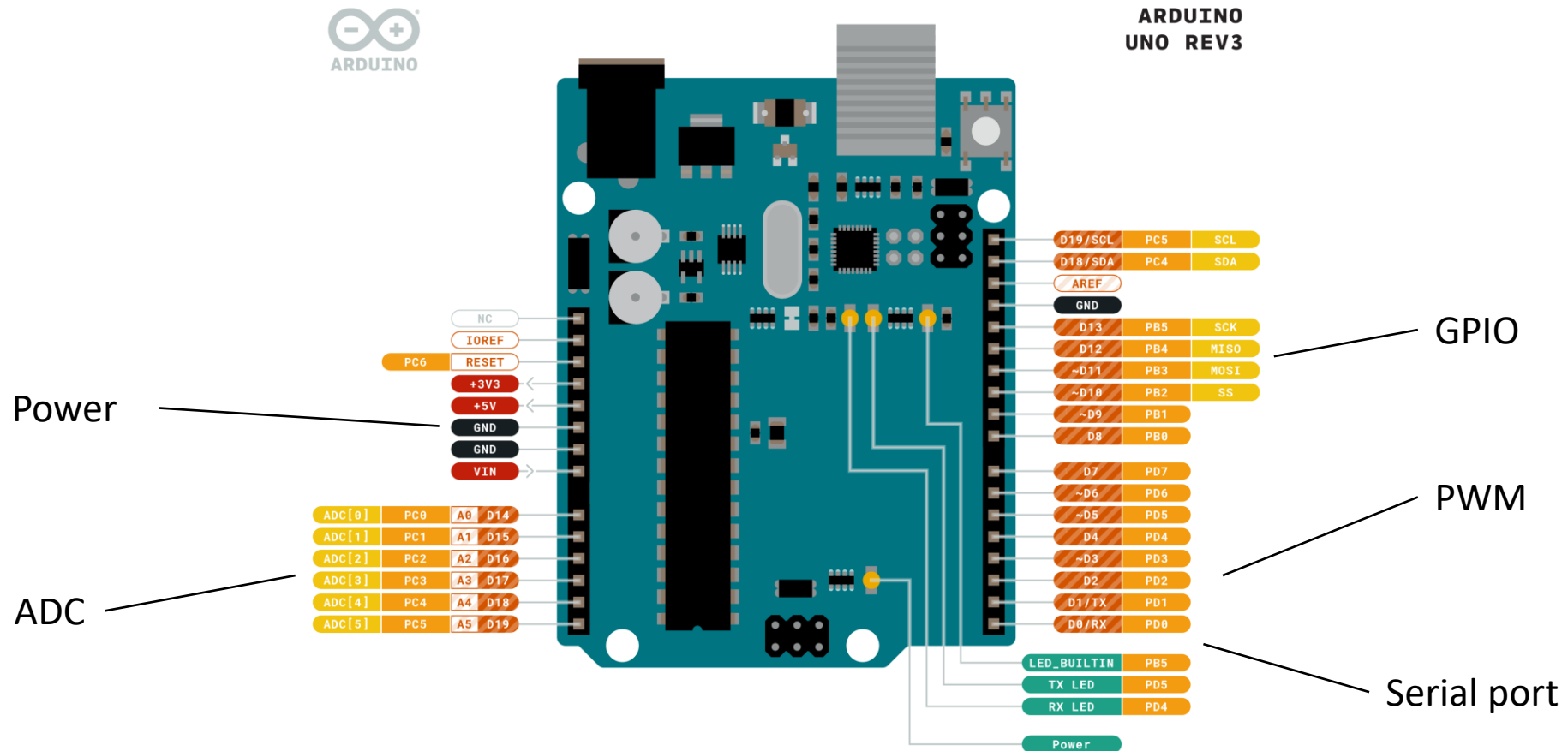
[Adafruit feather](#)

Small and powerful
arduino-like arduino, with
a lot of additional sensors
integrated

[Raspberry PI](#)

Not an arduino, more like a small
computer = great processing power
and GUI, but only GPIO is available

Arduino Uno Pinout

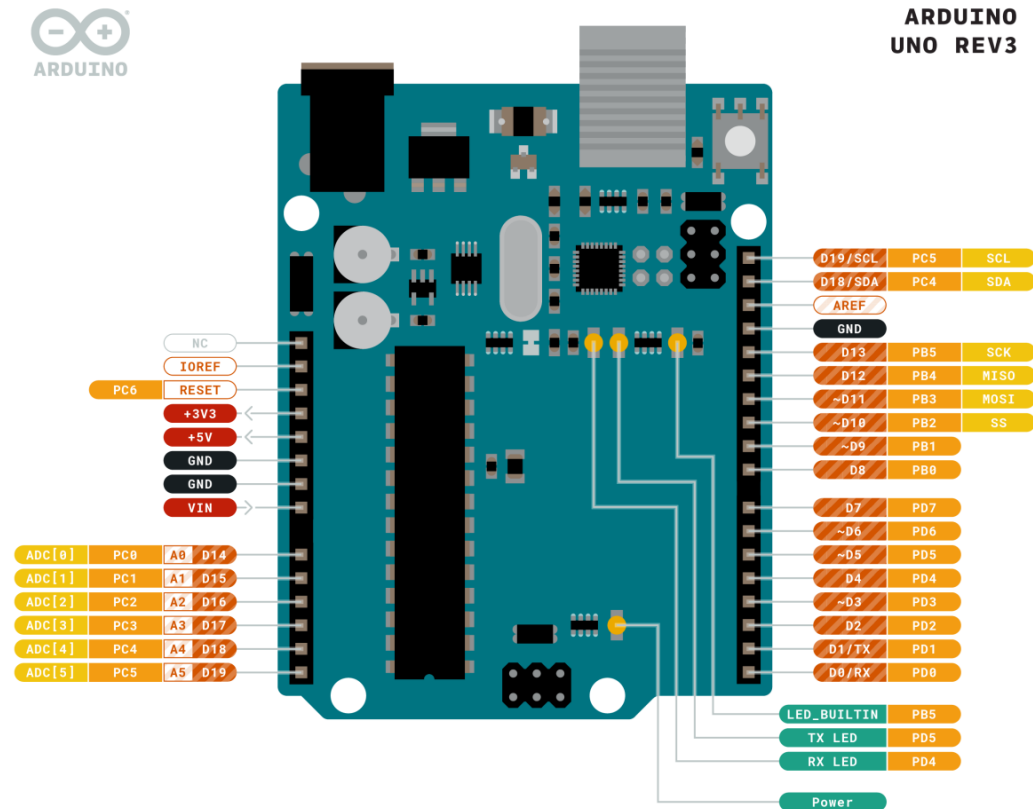


Ground	Internal Pin	Digital Pin	Microcontroller's Port
Power	SWD Pin	Analog Pin	
LED	Other Pin	Default	

ARDUINO.CC

This work is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/4.0/> or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

Arduino Uno specifications



Features:

- ADC
- UART (serial interface)
- Interrupts
- PWM
- GPIO
- ...

Ground	Internal Pin	Digital Pin	Microcontroller's Port
Power	SWD Pin	Analog Pin	
LED	Other Pin	Default	

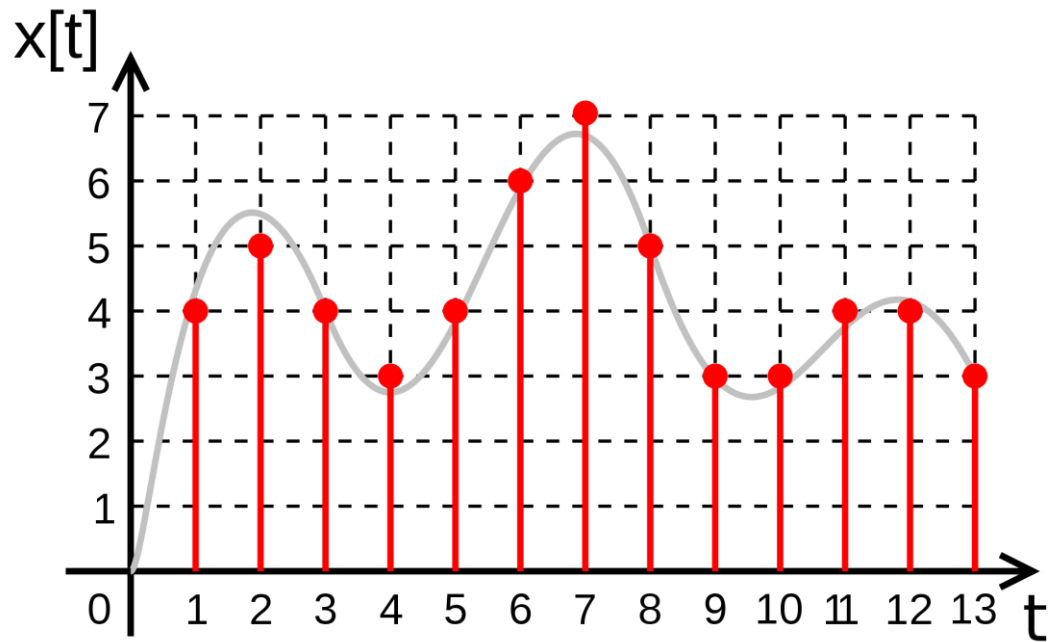
ARDUINO.CC



This work is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/4.0/> or send a letter to Creative Commons, PO Box 1888, Mountain View, CA 94042, USA.

Analog to Digital Converter

ADC, or Analog-to-Digital Converter, translates analog signals into digital values for microcontrollers like Arduino to read.



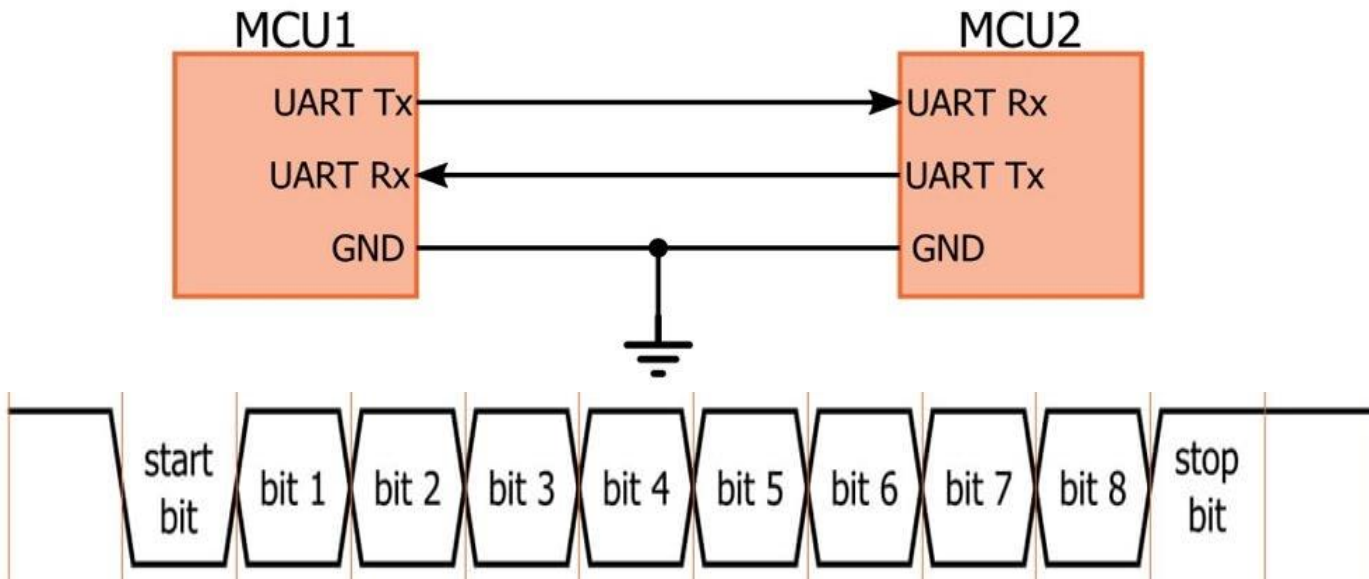
$$\text{Voltage} = \text{ADC reading} \times \frac{\text{Arduino voltage}}{\text{ADC Resolution}}$$

5V for Uno

$2^n - 1$ with n the ADC bits

Universal Asynchronous Reception and Transmission(UART)

UART (Universal Asynchronous Receiver/Transmitter) is a hardware communication protocol used for asynchronous serial communication between devices.



Key Terms

- **Start bit:** The first bit of a one-byte UART
- **Stop bit:** The last bit of a one-byte UART transmission.
- **Baud rate:** The approximate rate (in bits per second, or bps) at which data can be transferred.

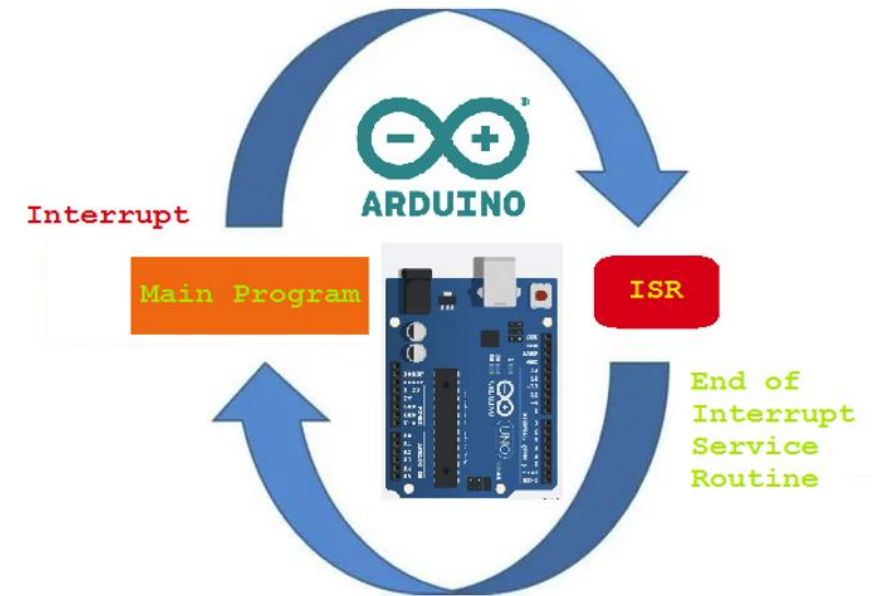
Interrupts

- **What is an Interrupt?**

- A signal to the CPU to pause and handle external events, enhancing real-time responsiveness and computational efficiency.

- **Key Concepts**

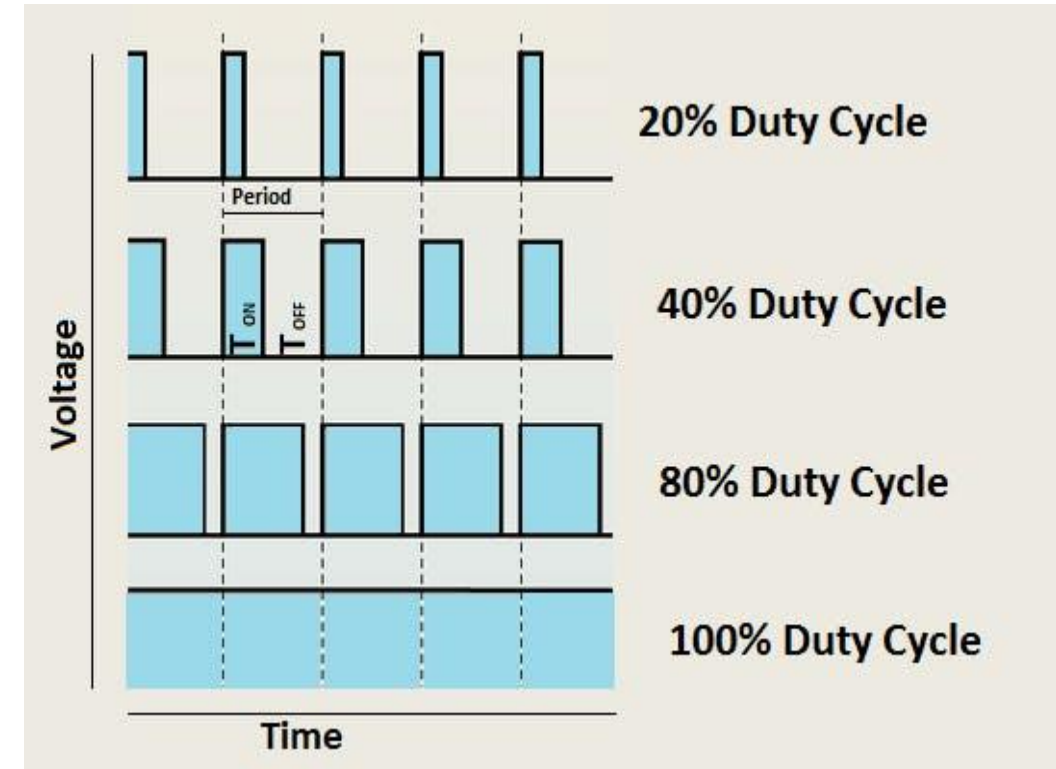
- **Types:** Hardware and Software Interrupts.
- **Purpose:** Ensure timely system responses and allow multitasking during I/O operations.
- **Usage:** Employed to handle tasks like responding to a button press, receiving data, or managing timed events in embedded systems.



Pulse Width Modulation (PWM)

PWM stands for Pulse Width Modulation and it is a technique used in controlling the brightness of LED, speed control of DC motor, controlling a servo motor or where you have to get analog output with digital means.

- **TON (On Time):** It is the time when the signal is high.
- **TOFF (Off Time):** It is the time when the signal is low.
- **Period:** It is the sum of on time and off time.
- **Duty Cycle:** It is the percentage of time when the signal was high during the time of period.



Arduino Uno specifications

Specs are available on the [arduino store](https://www.arduino.cc/en/Main/arduinoBoardUno)

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA



Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Select the peripherals



Arduino Uno

DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V



ECON Servo-180°

Dimensions	40.8*20.1*38mm
Product Weight	44g
Output Shaft Style	25T
Angle Rotation	180 degree
Voltage Range	4.8~7.2 Volts
No-Load Speed(6.0V)	0.19sec/60°
No-Load Speed(4.8V)	0.23sec/60°
Stall Torque(6.0V)	4.1kg.cm
Stall Torque(4.8V)	3.2kg.cm
Max PWM Signal Range(Standard)	500~2500usec
Travel per μ s(out of box)	/
Max Travel(out of box)	/
Pulse Amplitude	3~5V
Operating Temperature	(-)10 to +50 degree C
Current Drain -idle(6.0V)	20mA
Current Drain - no-load(6.0V)	200mA

sensors

What is the robot current configuration ?

- Joints position
- Speed
- Power consumption
- Heat

What is applied on the environment ?

- Force
- Contact

Localise the robot:

- GPS
- Vision

Interact:

- With the environment (ex: move an object)
- With the human (ex: assistive Robotics)

Robotic sensors are used to estimate a robot's condition and environment. These signals are passed to a controller to enable appropriate behaviour.

Most of the time, the controller is an algorithm implemented on a microcontroller (ex: Arduino) that controls the actuators depending on the sensor data



Arduino sensors

Sensor classification

By working principle:

- Resistive
- Capacitive
- Magnetic
- Optical
- MEMS
- Sound
- ...

By function:

- *Vision and Imaging Sensors*
- *Temperature Sensors*
- *Radiation Sensors*
- *Proximity Sensors*
- *Position Sensors*
- *Motion Sensors*
- *Photoelectric Sensors*
- *Particle Sensors*
- *Metal Sensors*
- *Pressure Sensors*
- *Level Sensors*
- *Leak Sensors*
- *Humidity Sensors*
- *Gas and Chemical Sensors*
- *Force Sensors*
- *Flow Sensors*
- *Flaw Sensors*
- *Flame Sensors*
- *Electrical Sensors*
- *Contact Sensors*
- *Non-Contact Sensors*

By return values : digital or analog

Position sensing summary

Resistances

- For small motion range
- Should be in contact with the moving part
- Rotational and Linear
- Continuous



A typical single-turn potentiometer

Encoders

- For small and large range of motion
- Should be in contact with the moving part
- Rotational and Linear
- Discrete



*Incremental optical encoder
Broadcom*

Distance

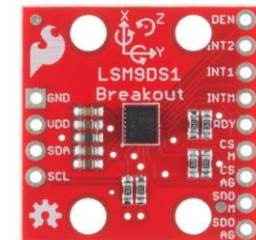
- Contactless sensing
- To choose depending of the object characteristics
- Continuous or discrete



Ultrasonic sensor

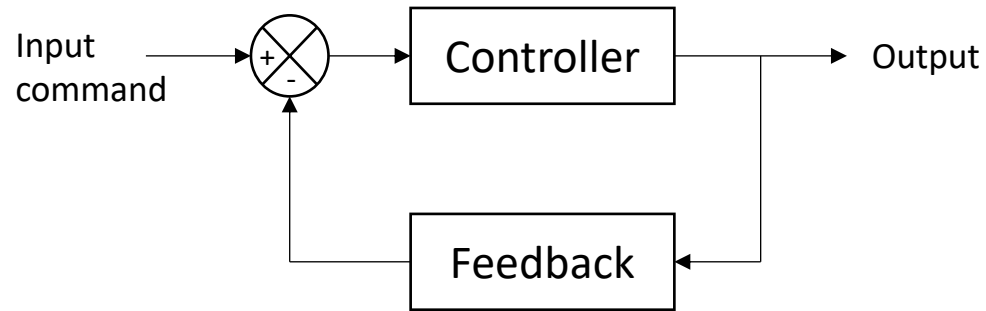
Acceleration/speed

- Easily integrable
- Requires integration to access the position

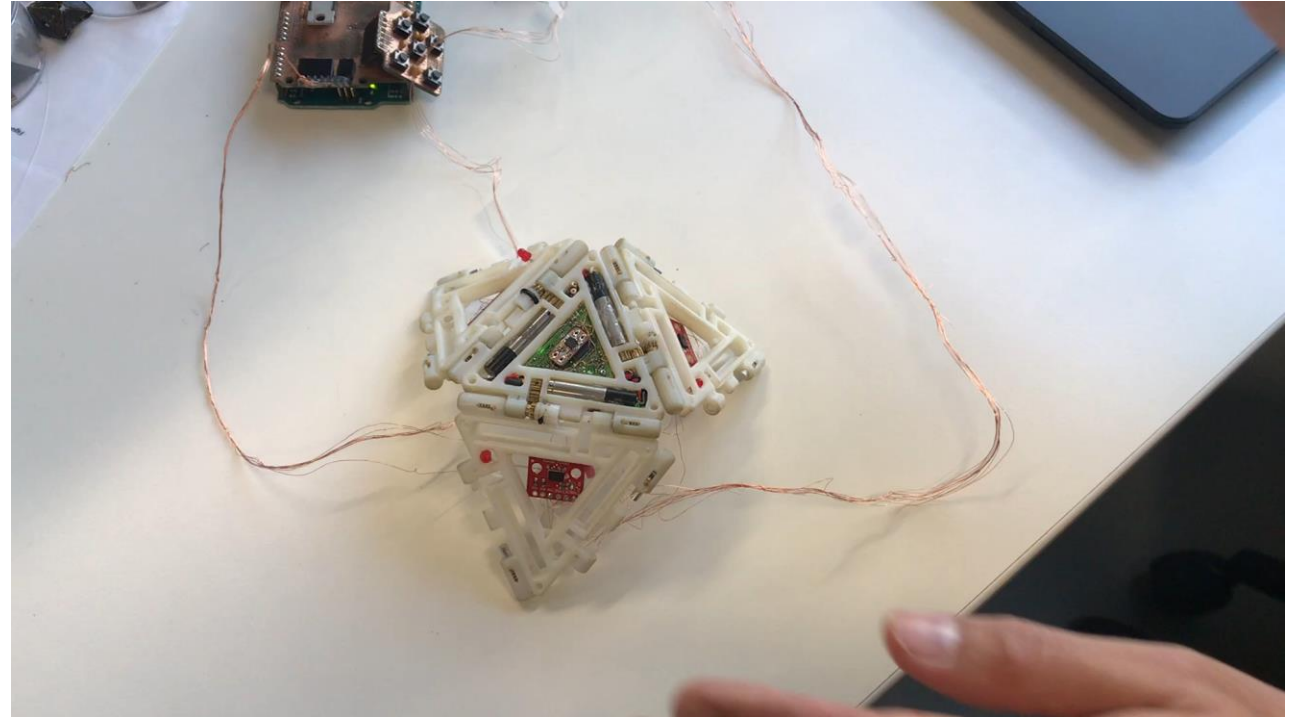


*9DoF-IMU-Breakout
SparkFun*

Application example



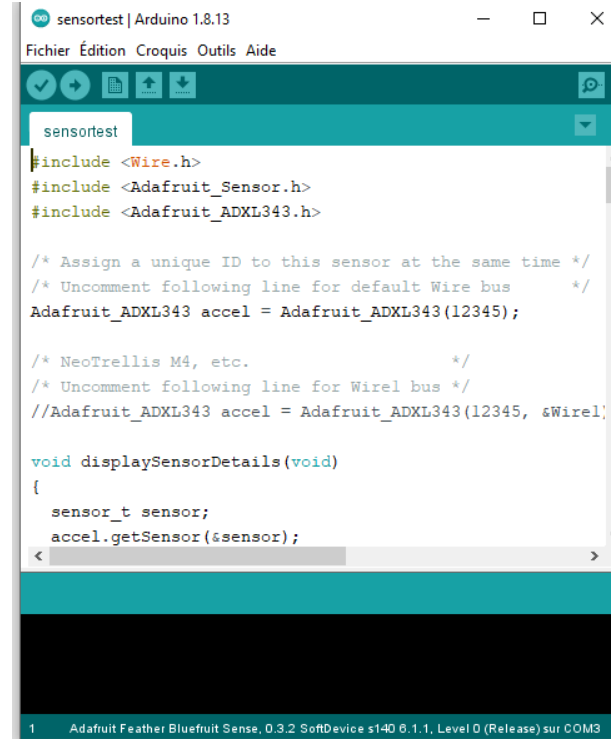
- Inertial Measurement Unit give the robot's configuration
- Motors' relative rotation angles with encoders
- Interaction with human through distance sensors on the legs



Huang, J. L . et al. (2018). A reconfigurable interactive interface for controlling robotic origami in virtual environments. *The International Journal of Robotics Research*, 37(6), 629-647.

Basics in Arduino programming: IDE

[Download basic usual IDE](#)

A screenshot of the Arduino IDE interface. The title bar shows "sensorTest | Arduino 1.8.13". The menu bar includes "Fichier", "Édition", "Croquis", "Outils", and "Aide". The toolbar has icons for opening, saving, and running. The main text area shows the following code:

```
sensorTest
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_ADXL343.h>

/* Assign a unique ID to this sensor at the same time */
/* Uncomment following line for default Wire bus */
Adafruit_ADXL343 accel = Adafruit_ADXL343(12345);

/* NeoTrellis M4, etc. */
/* Uncomment following line for WireL bus */
//Adafruit_ADXL343 accel = Adafruit_ADXL343(12345, &WireL);

void displaySensorDetails(void)
{
  sensor_t sensor;
  accel.getSensor(&sensor);
}
```

The status bar at the bottom indicates "1 Adafruit Feather Bluefruit Sense, 0.3.2 SoftDevice s140 0.1.1, Level 0 (Release) sur COM3".

Pros:

- Perfect for beginners
- Easy to use
- Great Serial plotter
- GUI makes it easy to change the board connection parameters

Cons:

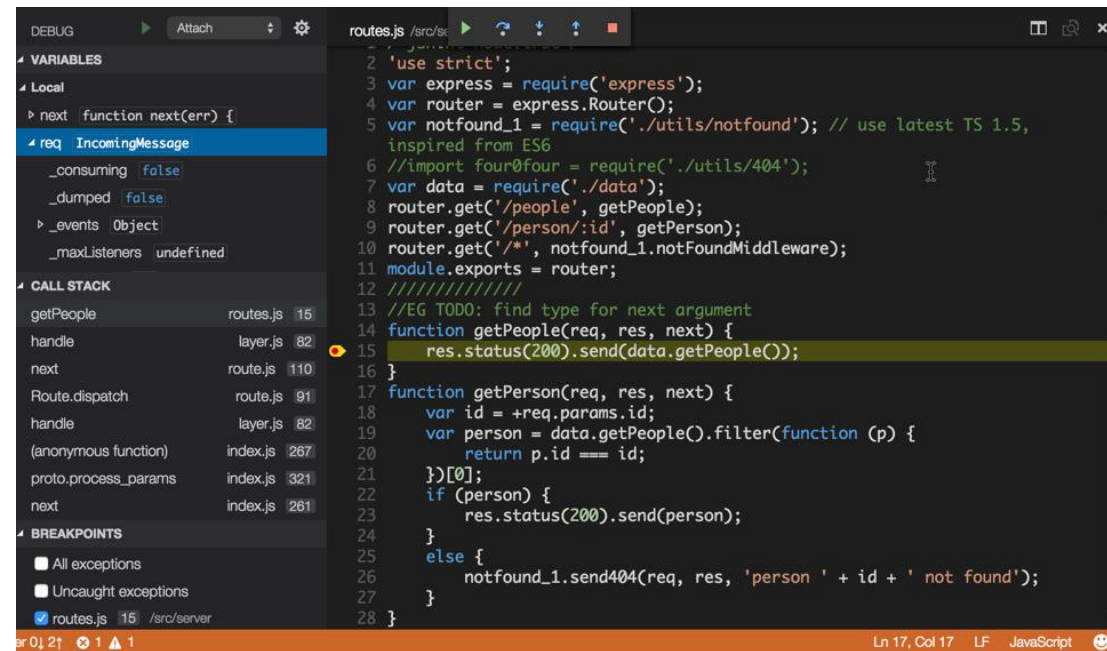
- No color highlight
- No autocomplete
- Difficult Multi-file programming
- It's like programming on a notepad

Basics in Arduino programming: IDE

Visual studio code

+

PlatformIO



Pros:

- For **real** programmers
- Multi-file organisation
- Programs easily transportable
- DARK MODE
- Works with a lot of boards

Cons:

- Serial communication extremely basic
- Not as straightforward to use than the official IDE

Arduino is C++

(So you can use any C++ IDE)

Basics in Arduino programming

Arduino is embedded C made easy

Arduino code must
contains two functions:

Run once at the code
initialization (board startup)

```
void setup(void)
{
}
```

Init

- Variables
- GPIO
- Timers
- Interrupts
- All the system parameters
your program needs

Infinite loop, equivalent
to while(true)

```
void loop(void)
{
}
```

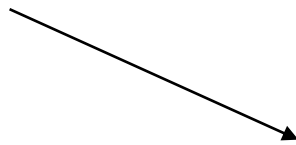
Run continuously while the
board is power supplied

- Read sensors
- Send control commands
- Communicate with the
computer
- Contains the program logics

Basics in Arduino programming

Example

*Declare variables as global
(variables declared inside a function are
deleted when the function ends)*



```
/*  
DigitalReadSerial
```

```
  
Reads a digital input on pin 2, prints the result to the  
Serial Monitor
```

```
  
This example code is in the public domain.  
http://www.arduino.cc/en/Tutorial/DigitalReadSerial  
*/
```

```
  
// digital pin 2 has a pushbutton attached to it.  
Give it a name:  
int pushButton = 2;
```

```
  
// the setup routine runs once when you press  
reset:  
void setup() {  
  // initialize serial communication at 9600 bits  
  per second:  
  Serial.begin(9600);  
  // make the pushbutton's pin an input:  
  pinMode(pushButton, INPUT);  
}
```

```
  
// the loop routine runs over and over again  
forever:  
void loop() {  
  // read the input pin:  
  int buttonState = digitalRead(pushButton);  
  // print out the state of the button:  
  Serial.println(buttonState);  
  delay(1);    // delay in between reads for  
  stability  
}
```

Lots of examples are available under *File* ->
Examples In the Arduino IDE

Basics in Arduino programming

Functions

Please look at the [datasheet](#) before using built-in functions

Declaration → `void customFunction(char inputVariable);`

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

Call → `customFunction('a');`

Function body → `void customFunction(char inputVariable){
 Serial.println(inputVariable);`

ME-410 Mechanical Product Design & Development

EPFL Demo: Programming the Uno Using the Arduino IDE

