Connected Things

COMP5047 – Lecture 04

Anusha Withana

The School of Computer Science
The University of Sydney

Is everything connected?

- Hard to find standalone systems
  - E.g. Electric razor, blow dryer, smoke alarm

- Even then, you will find there are interconnected elements inside
  - E.g. controllers, switches, motors, sensors

- Devices need information exchange
  - Send or receive data

How do devices communicate?

How to create smart objects by connecting sensors/actuators with computing elements (e.g. microcontrollers) and then connect them to networks.

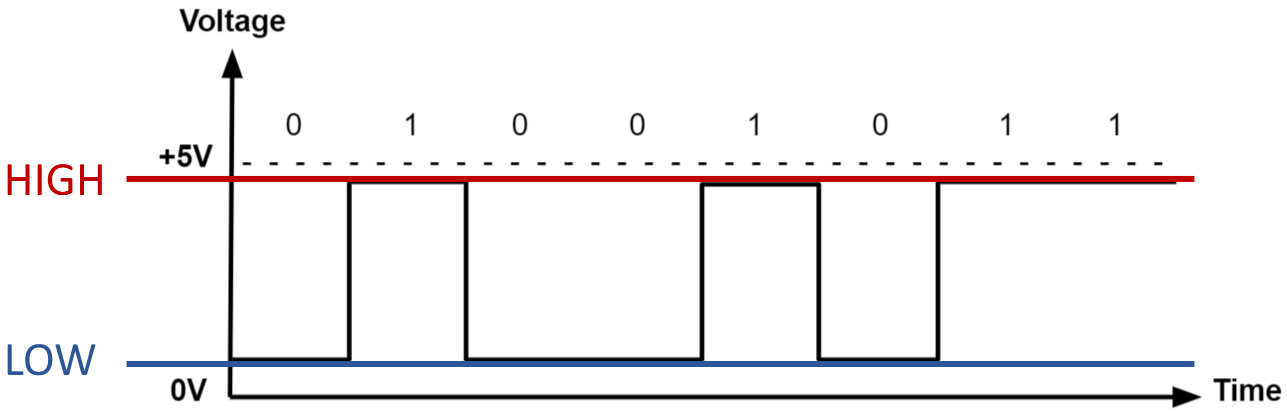
Digital

(of signals or data) expressed as series of the digits 0 and 1, typically represented by values of a physical quantity such as voltage or magnetic polarization.

Definitions from Oxford Languages

Digital

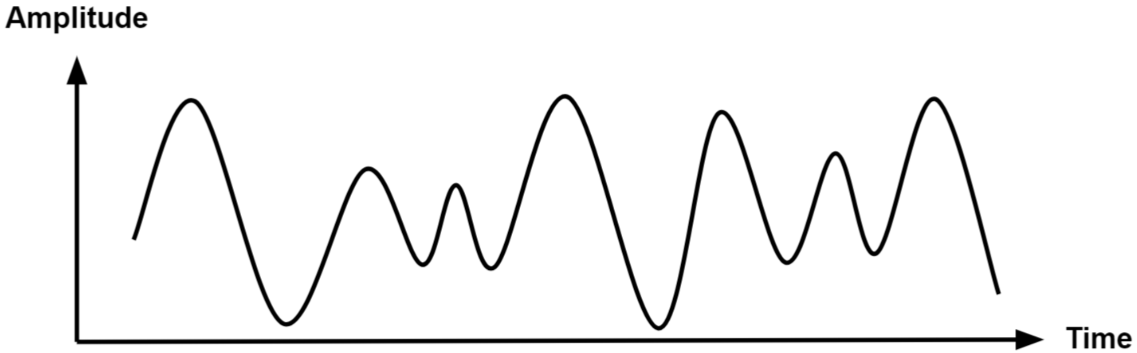
- Discrete signals
  - Amplitude
  - Time
- Represent numbers
  - Usually binary numbers
  - And we can use them to compute (+, -, *,..)
- Used in your computer
  - And your microcontroller
  - Various ranges of voltages
    - (0,5V), (0,3.3V), (0,1.8V), (-15V,15V),



https://www.monolithicpower.com/en/analog-vs-digital-signal

Analog or Real World Signals

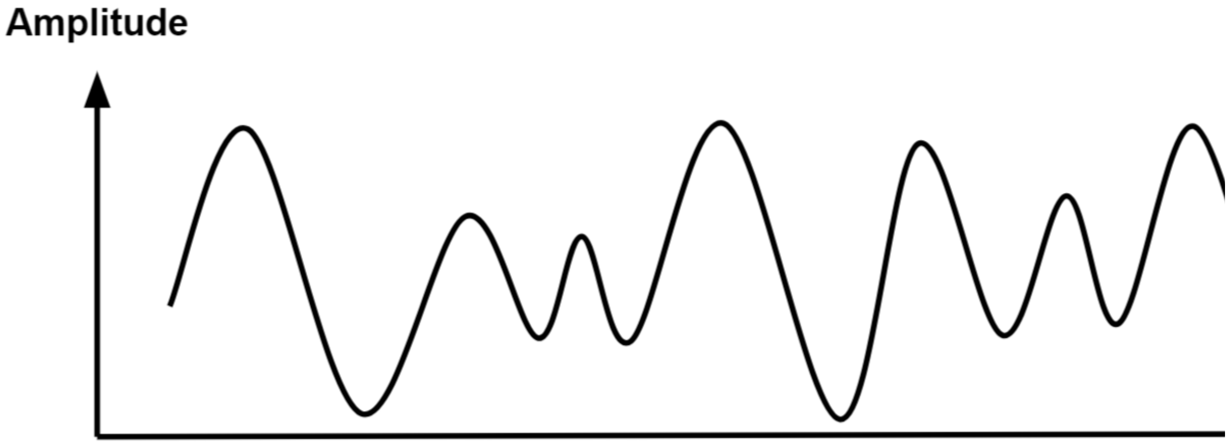
- Continuous signals
  - Amplitude
  - Time
- A lot of variations



https://www.monolithicpower.com/en/analog-vs-digital-signal

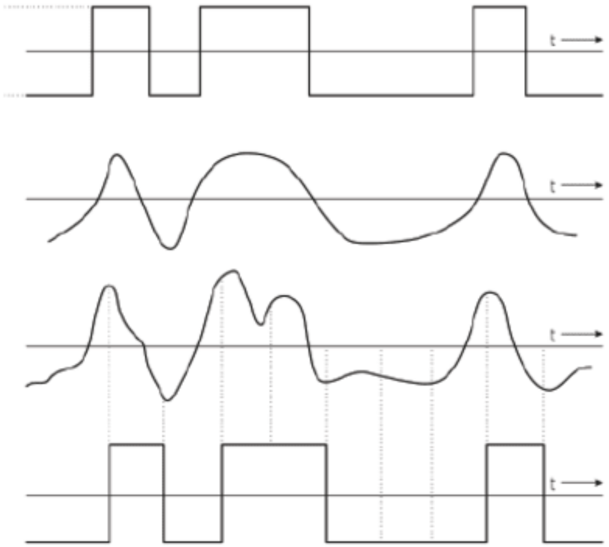
Analog Vs. Digital

- More likely to get affected by noise reducing accuracy
  - considerable observational errors
  - Network performance is subjected to deterioration by noise.



https://www.monolithicpower.com/en/analog-vs-digital-signal

- Less affected since noise response are analog in nature
  - has low observational errors.
  - Can be noise-immune during transmission and write/read cycle.



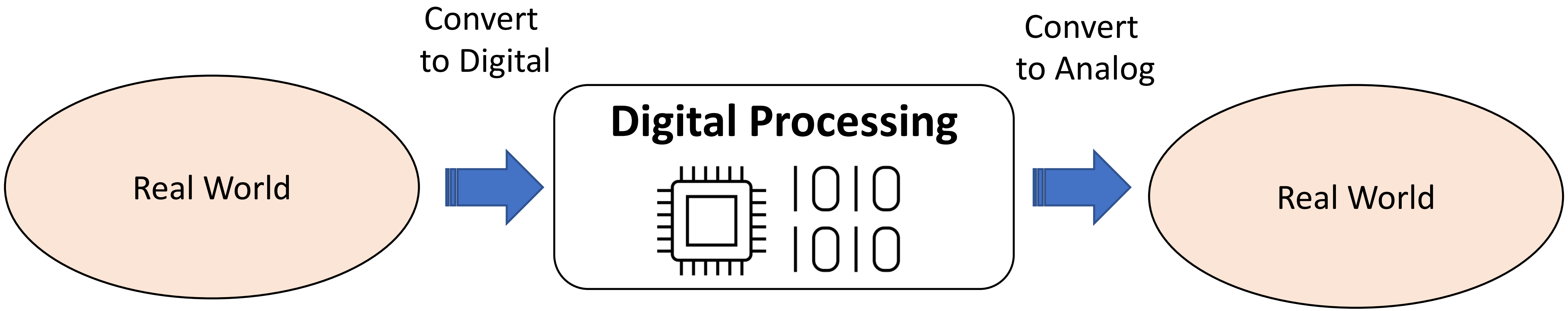
https://www.sigmadzn.com/2018/11/27/high-speed-digital-design-part-1/

Analog Vs. Digital

- More likely to get affected by noise reducing accuracy
    - considerable observational errors
    - Network performance is subjected to deterioration by noise during
- Analog instrument draws large power
- Analog signal processing can be done in real time and consumes less bandwidth.

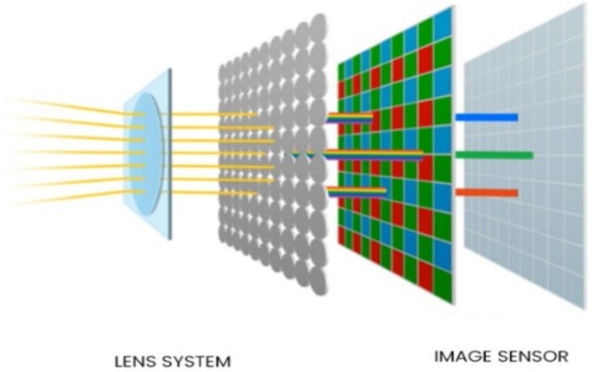
- Less affected since noise response are analog in nature
    - has low observational errors.
    - Can be noise-immune during transmission and write/read cycle.
- Draws comparatively much lower amount of power
- No guarantee that digital signal processing can be done in real time and consumes more bandwidth to carry out the same information.

Real World / Digital

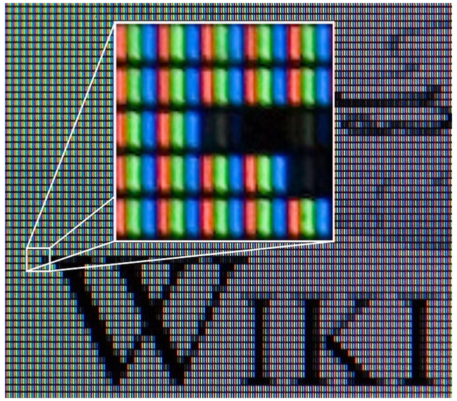


Convert to Digital

Convert to Analog



https://www.embedded.com/the-fundamentals-of-image-quality-tuning/



https://commons.wikimedia.org/

How do we connect?

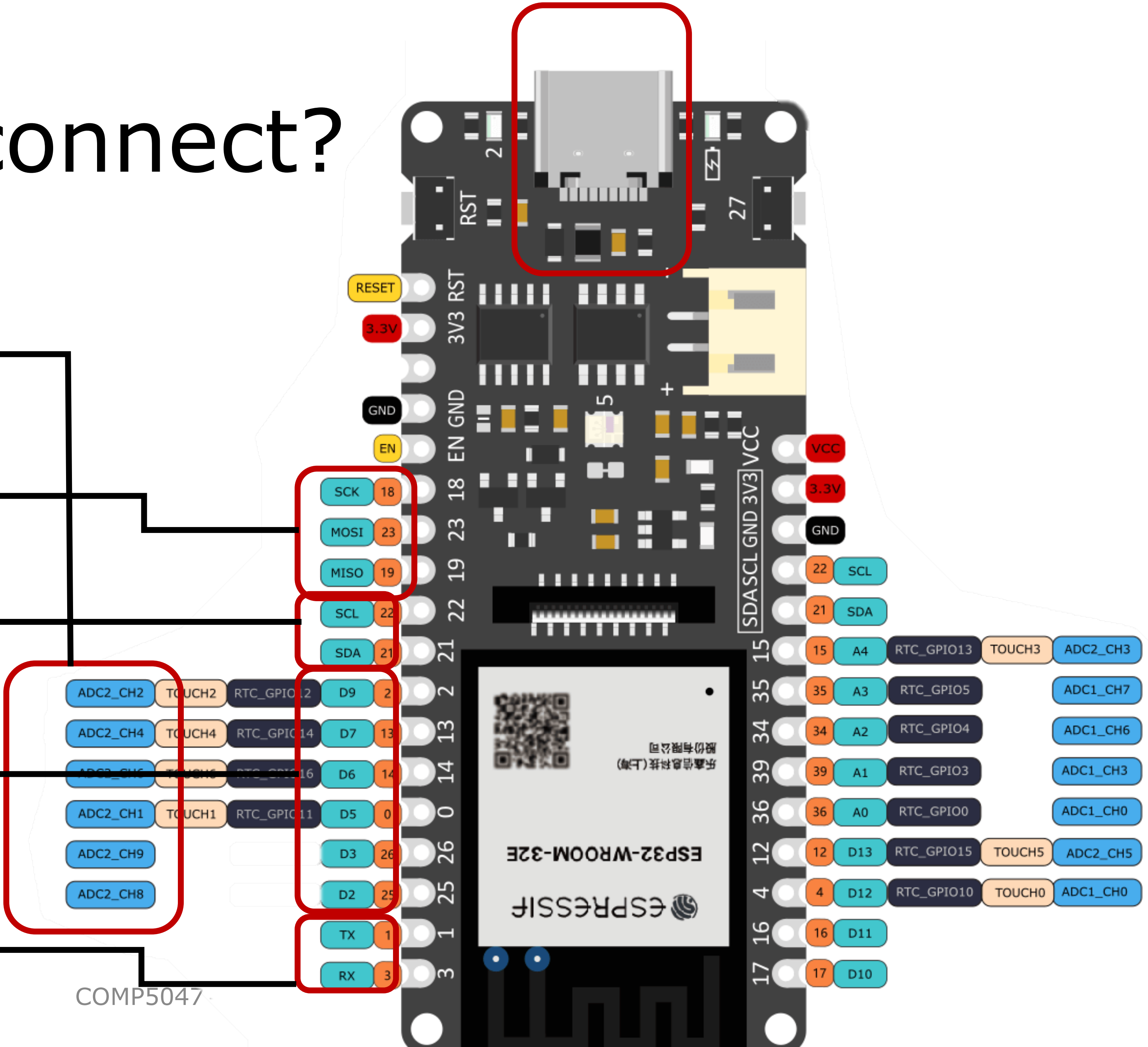
Analog to Digital Converter (**ADC**)

Serial Peripheral Interface (**SPI**)

I2C (Inter-Integrated Circuit, eye-squared-C)

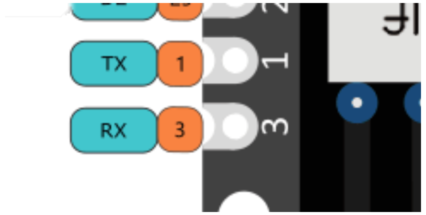
Digital IO / PWM

Universal Asynchronous Receiver/Transmitter (**UART**)/ **COM** port



UART/COM Port

- A serial interface
- It was widely used before USB
- A digital interface
  - Now commonly used in PerComp devices
  - E.g. Your µController has a serial ports (one through USB)
- Has so many variations
- Full duplex
  - Can transmit and receive at the same time

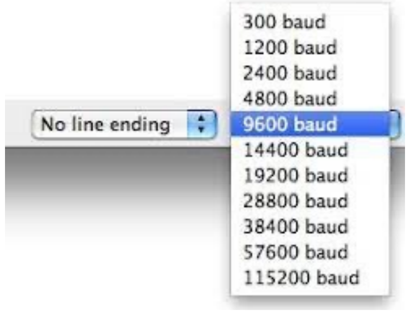


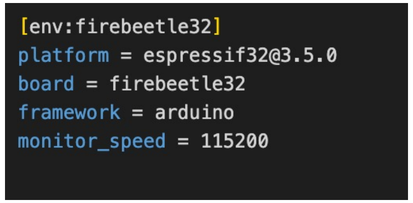
UART/COM Port

- Slower transmission rates
  - Typical max is is about 115 Kbps (kilobits per second)
- Shorter range (cable length)
- It is point to point
  - Typically no multipoint capability
- USB and Ethernet has taken over
  - But still used in small devices, industrial communications, instruments

UART/COM Port

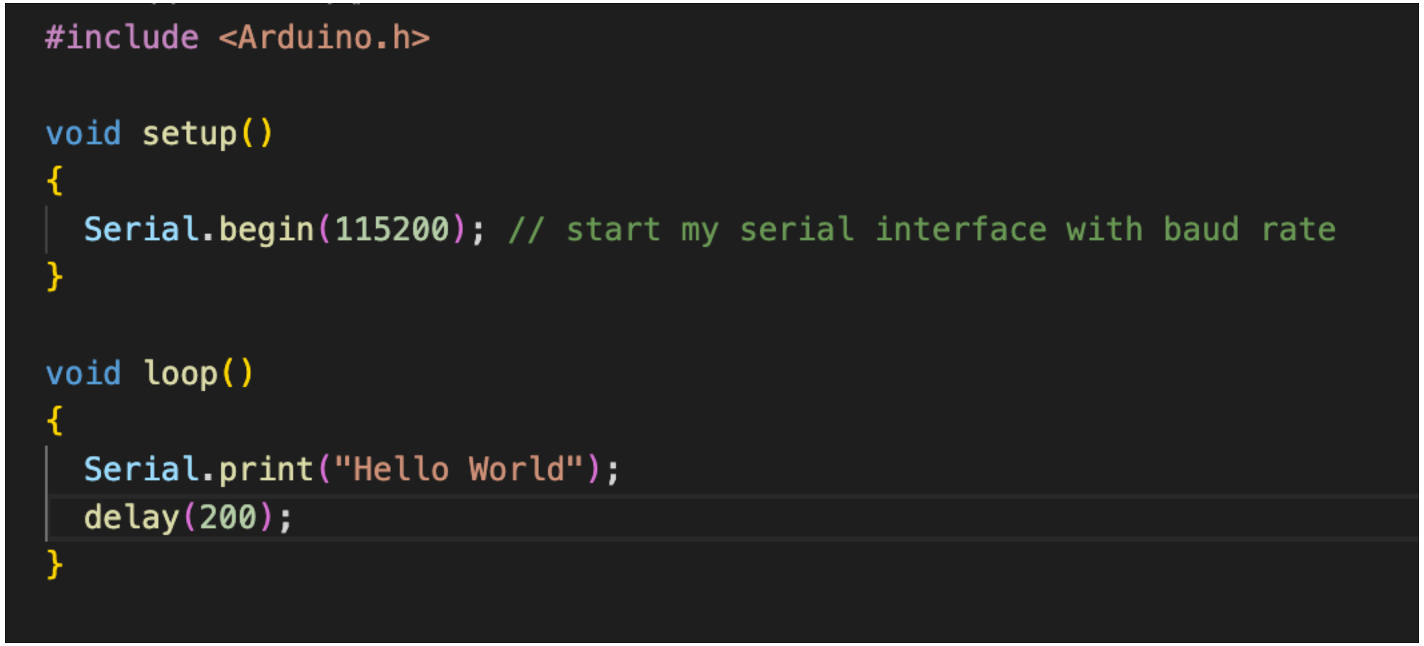
- Asynchronous
  - Two devices are not synced
  - You need to set a baud rate
  - The rate at which information is transferred
  - Higher the baud rate, more susceptible to noise



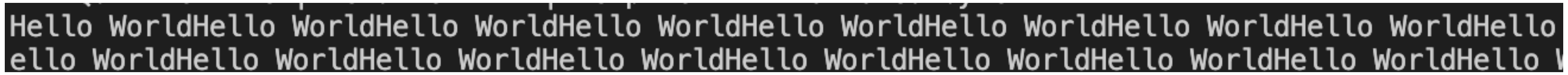


```
[env:firebeetle32]
platform = espressif32@3.5.0
board = firebeetle32
framework = arduino
monitor_speed = 115200
```

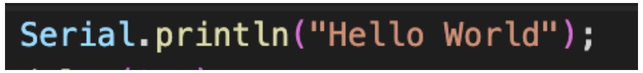
Let’s send some data from your µC



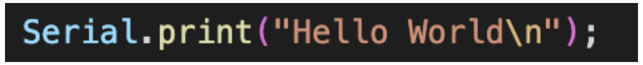
```
#include <Arduino.h>

void setup()
{
  Serial.begin(115200); // start my serial interface with baud rate
}

void loop()
{
  Serial.print("Hello World");
  delay(200);
}
```



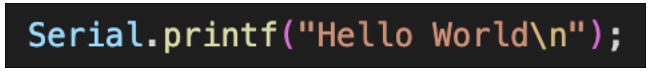
```
Hello WorldHello WorldHello WorldHello WorldHello WorldHello WorldHello WorldHello WorldHello
ello WorldHello WorldHello WorldHello WorldHello WorldHello WorldHello WorldHello WorldHello W
```



```
Serial.println("Hello World");
```



```
Serial.print("Hello World\n");
```

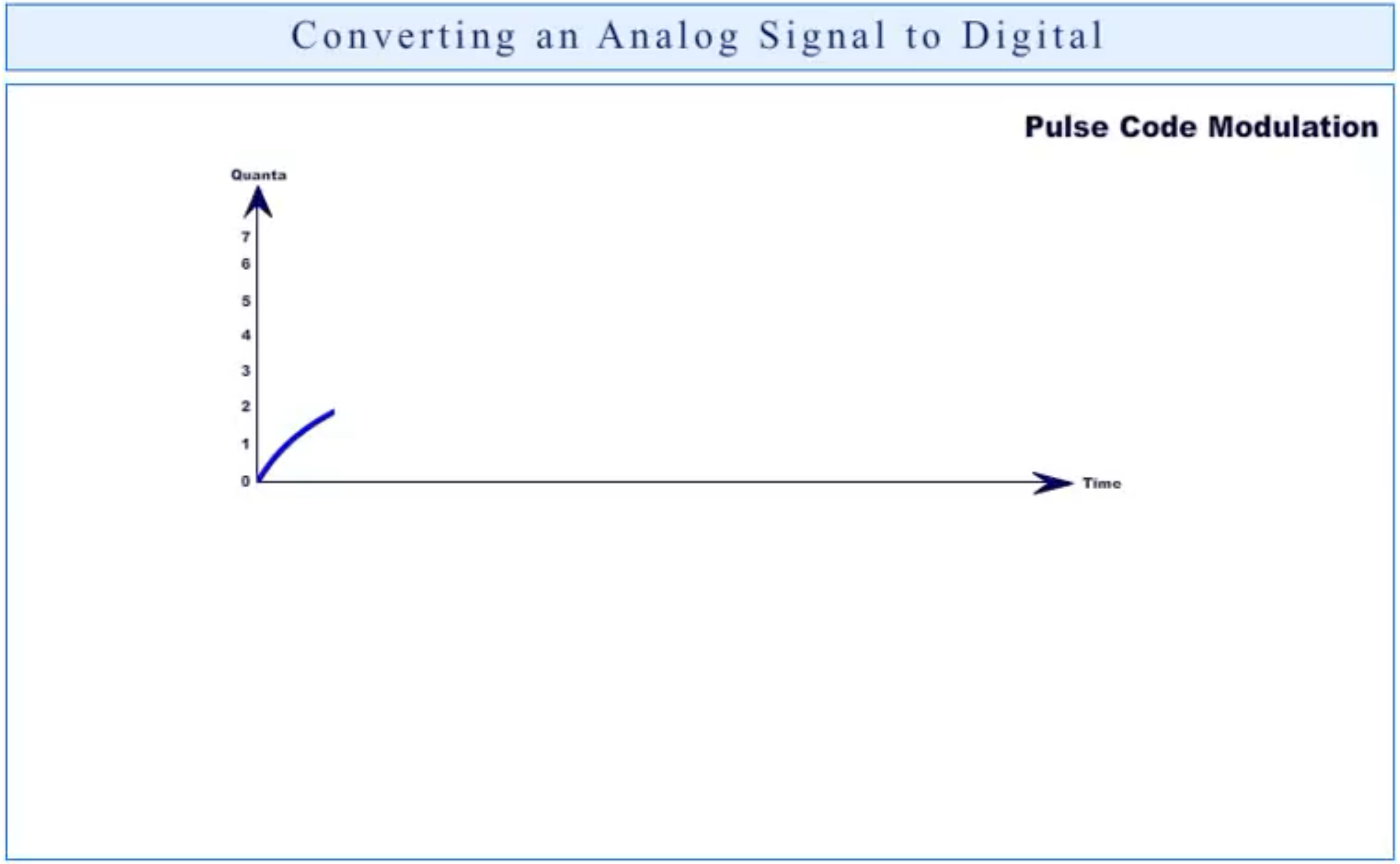


```
Serial.printf("Hello World\n");
```

Analog to Digital Converter (ADC)

- ADC
  - For analog band limited signals

1. Sample
2. Quantize
3. Encode



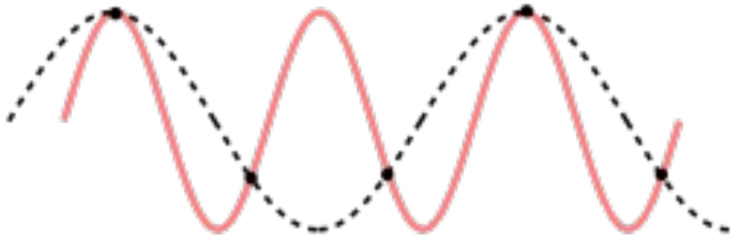
Channel: HowTo; https://www.youtube.com/watch?v=tZR7hLJx6Ms

Analog to Digital Converter (ADC)

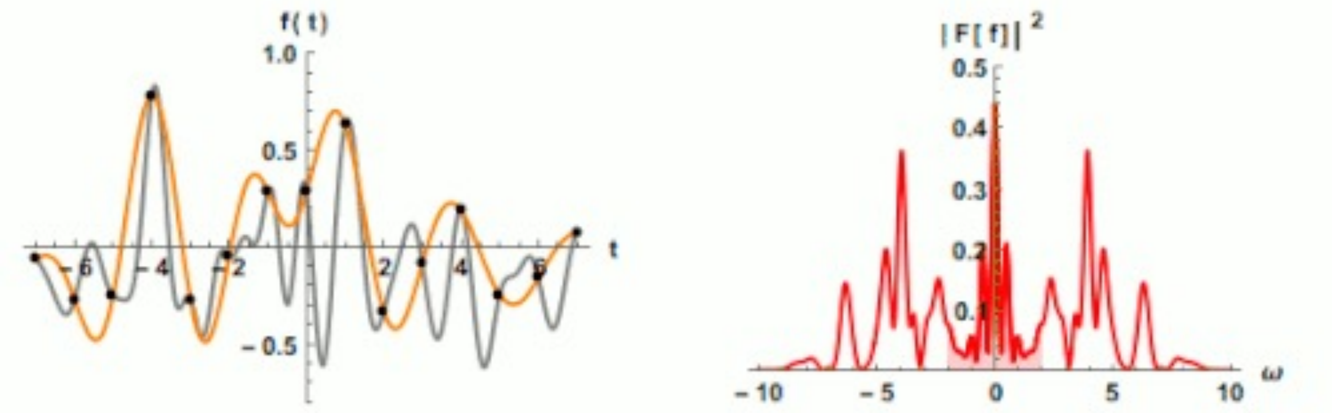
- Nyquist–Shannon sampling theorem

If a function *x(t)* contains no frequencies higher than *B* hertz (Hz), a sufficient sample-rate is therefore anything larger than *2B* samples per second.

Given sampling frequency f_s ; perfect reconstruction is guaranteed possible for a bandlimit $B < {f_s}/{2}$

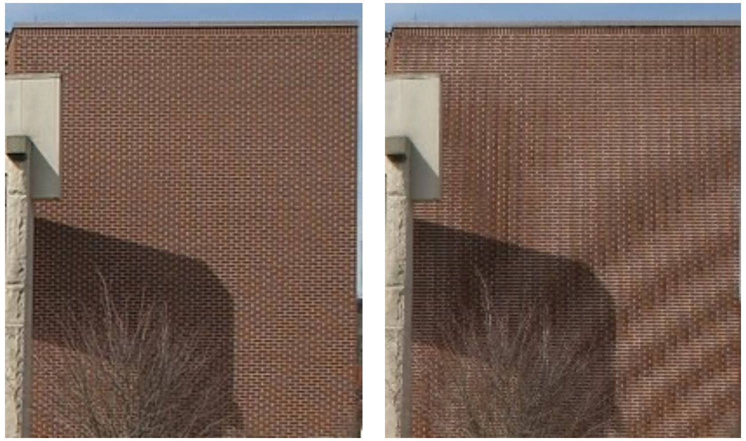


Adapted from Wikipedia



Analog to Digital Converter (ADC)

- Nyquist–Shannon sampling theorem
- Sub Nyquist sampling leads to aliasing
- If you sample too fast;
  - Too much data
  - A lot of power
  - Need expensive hardware



Subsampled image showing a Moiré pattern

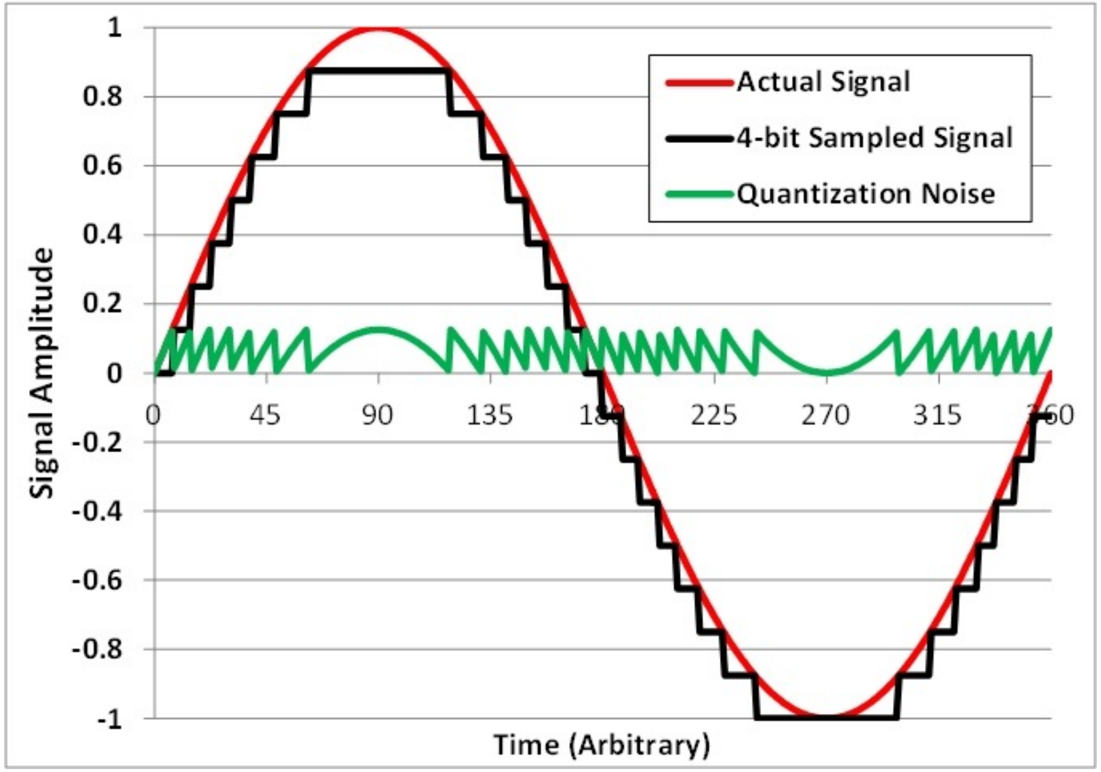
Adapted from Wikipedia

- You want to measure a fan rotor, rotating at $60 \pm 10Hz$, which sampling rate is best?
  - A) 20Hz, B) 60Hz, C) 200Hz, D)240Hz

Analog to Digital Converter (ADC)

- Quantization steps
- The resolution of ADC
    - Leads to quantization noise

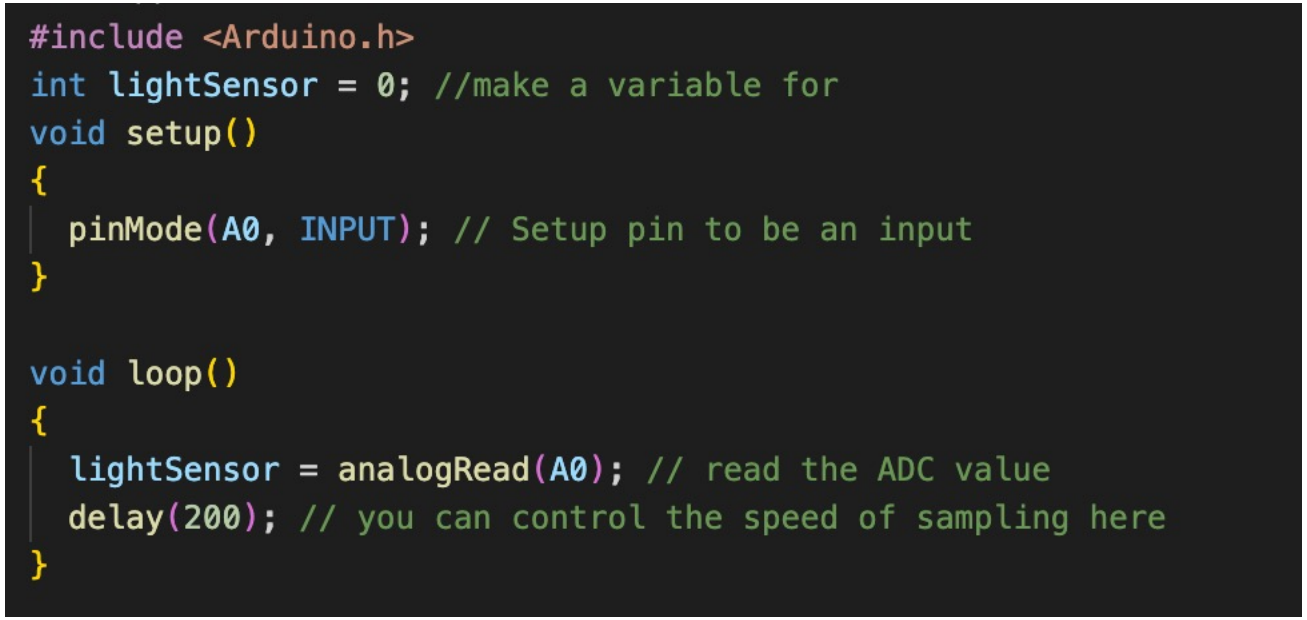
- Higher doesn’t mean better
    - Same issues as sampling rate



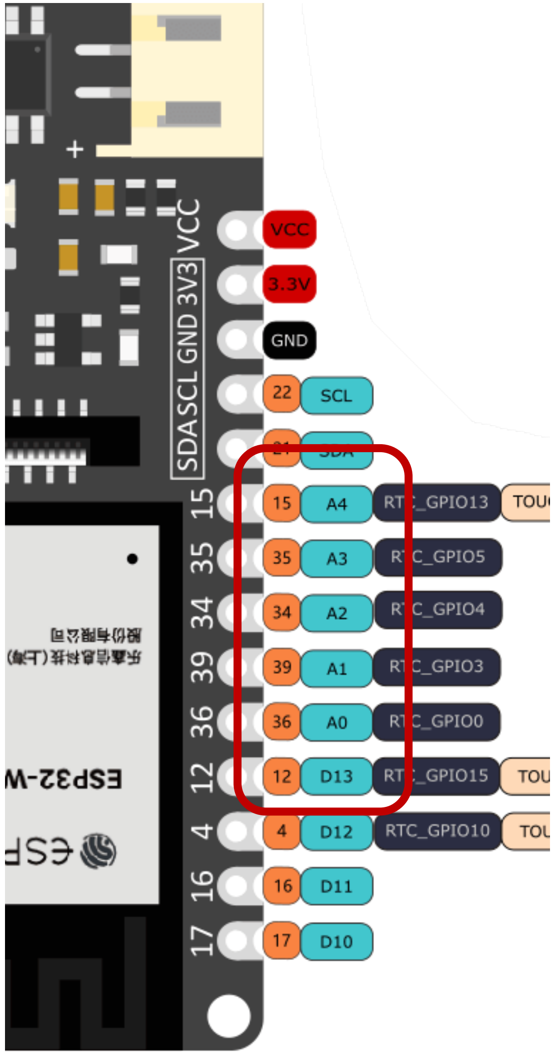
https://e2e.ti.com/

Analog to Digital Converter (ADC)

- How to?

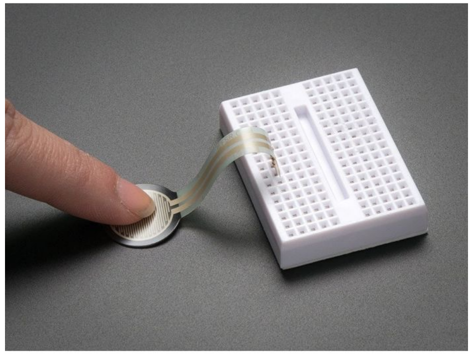


```
#include <Arduino.h>
int lightSensor = 0; //make a variable for
void setup()
{
  pinMode(A0, INPUT); // Setup pin to be an input
}

void loop()
{
  lightSensor = analogRead(A0); // read the ADC value
  delay(200); // you can control the speed of sampling here
}
```

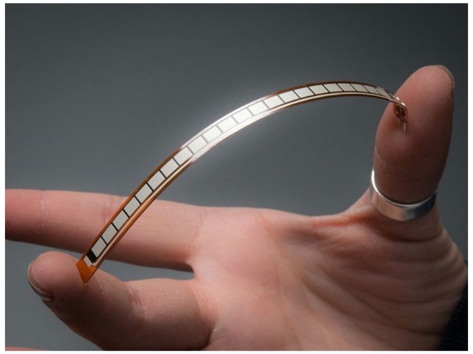


Analog to Digital Converter (ADC)

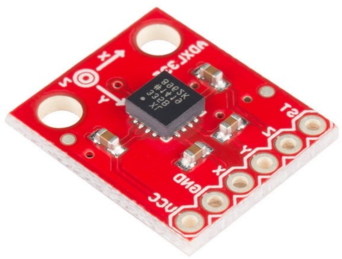
- Many sensors that gives an analog output



Force-Sensitive Resistor (FSR)



Flex sensor



Accelerometer - ADXL335

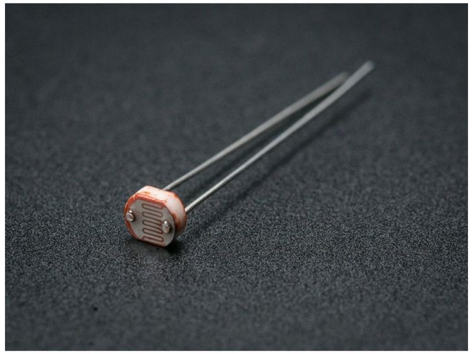
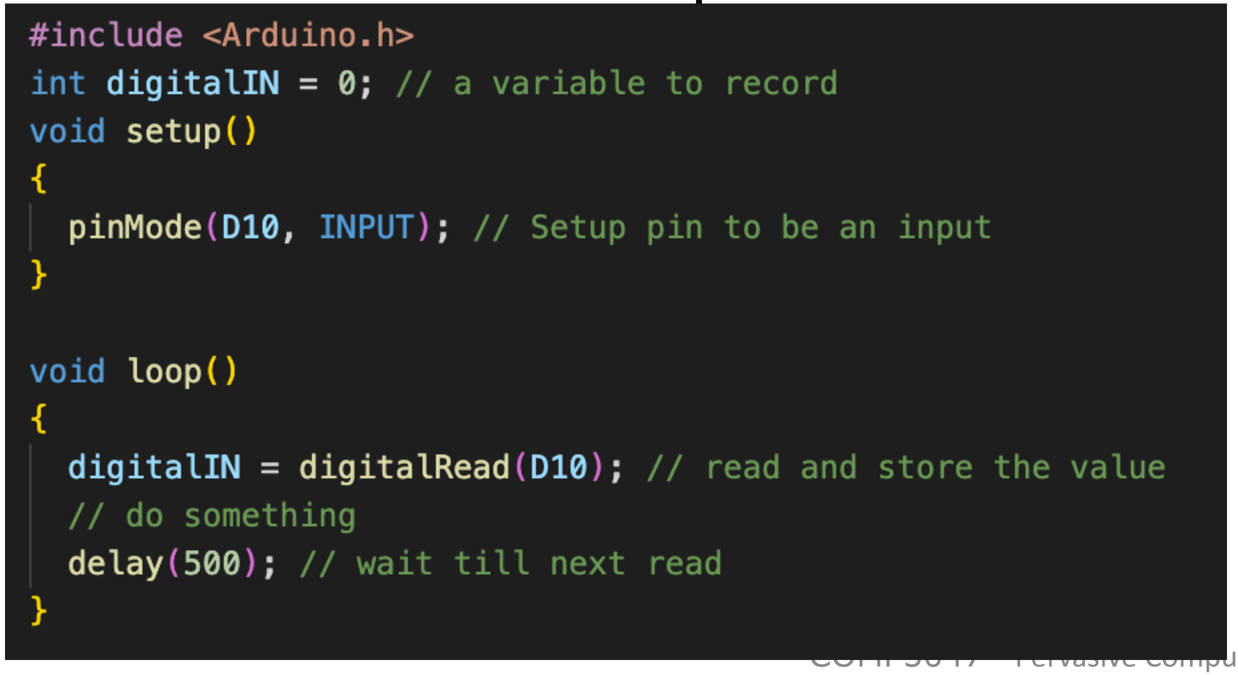


Photo cell (CdS photoresistor)

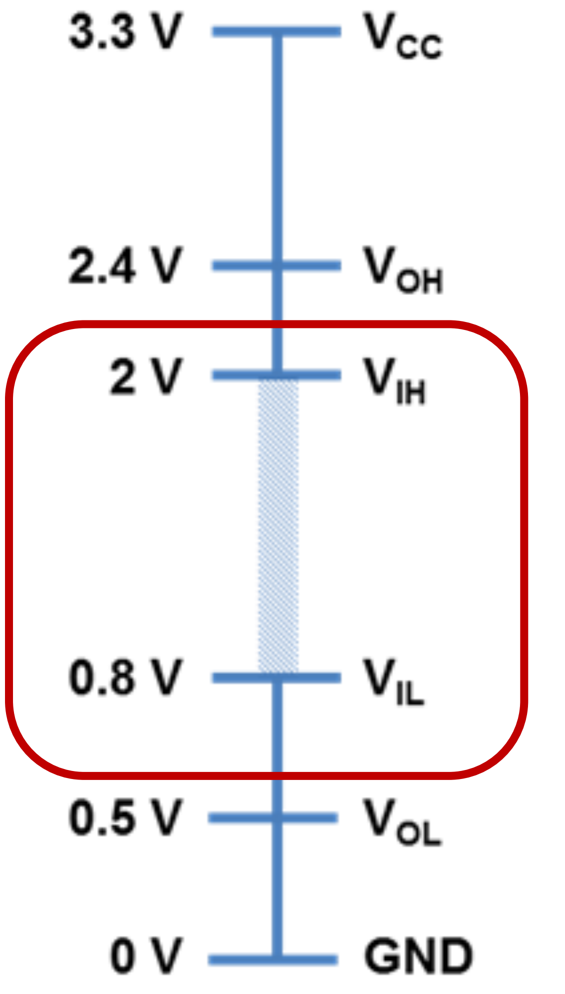
https://core-electronics.com.au/

Digital IO

- Input
  - Sense if a pin is **HIGH** or **LOW**
  - Depends on threshold voltages
  - Limited to fixed input



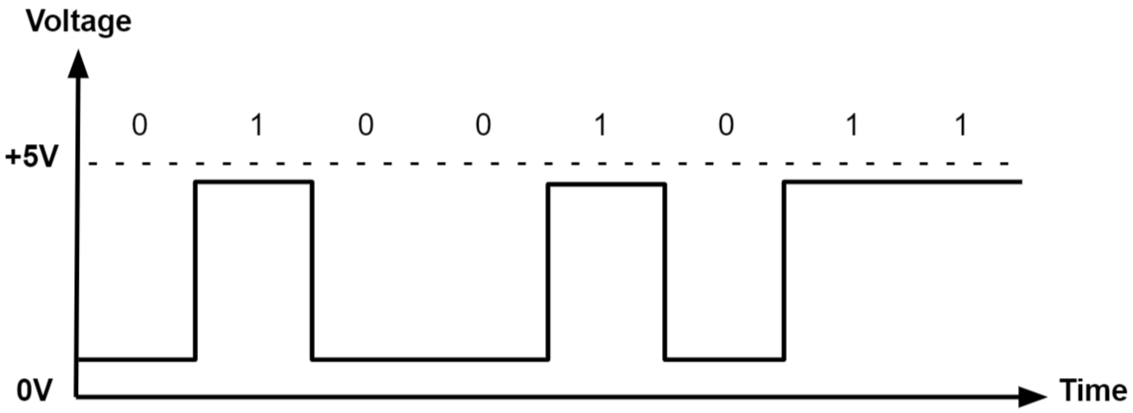
```
#include <Arduino.h>
int digitalIN = 0; // a variable to record
void setup()
{
  pinMode(D10, INPUT); // Setup pin to be an input
}

void loop()
{
  digitalIN = digitalRead(D10); // read and store the value
  // do something
  delay(500); // wait till next read
}
```



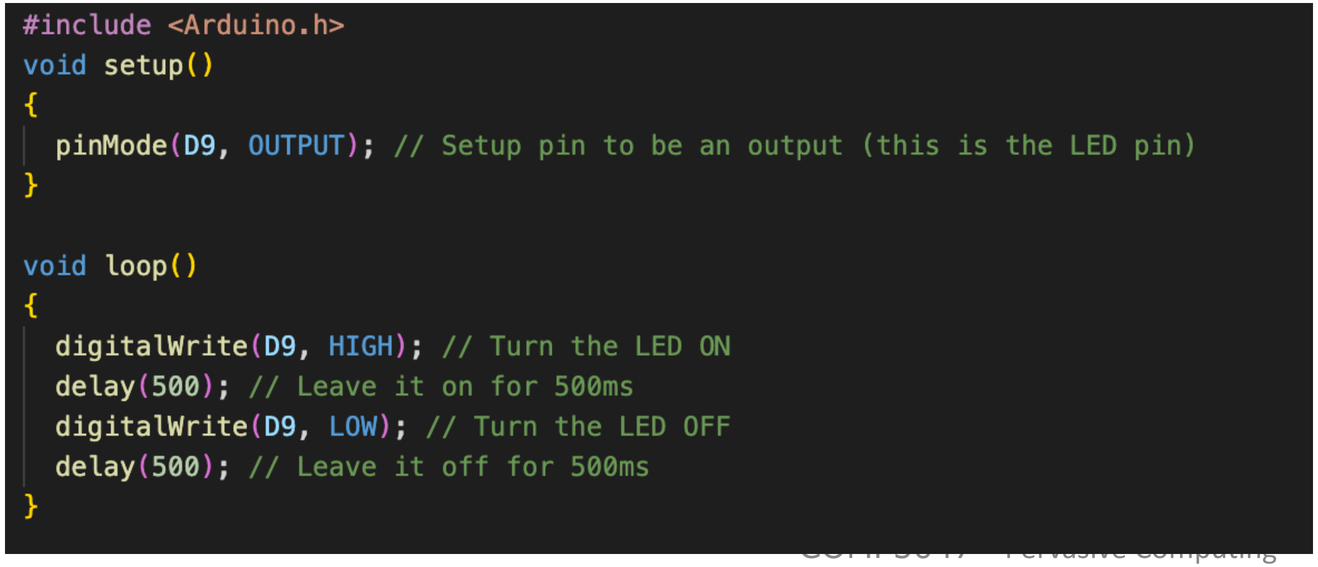
3.3 V Logic Families
(based on 74LVT04 Hex Inverter)

Digital IO

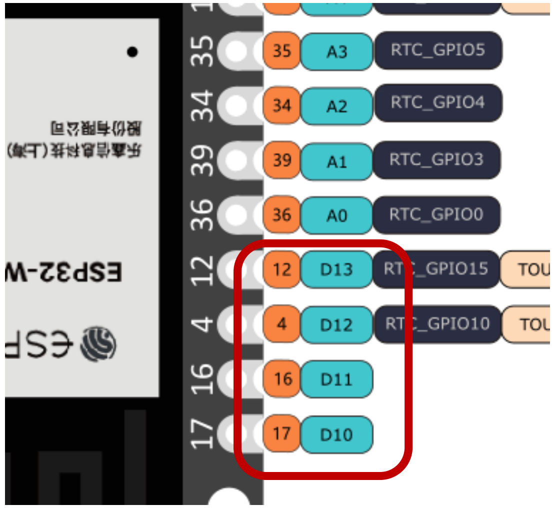
- Output
  - Set pin to **HIGH** or **LOW**
  - Actual voltage depends on the device
  - Limited to fixed output



https://www.monolithicpower.com/en/analog-vs-digital-signal



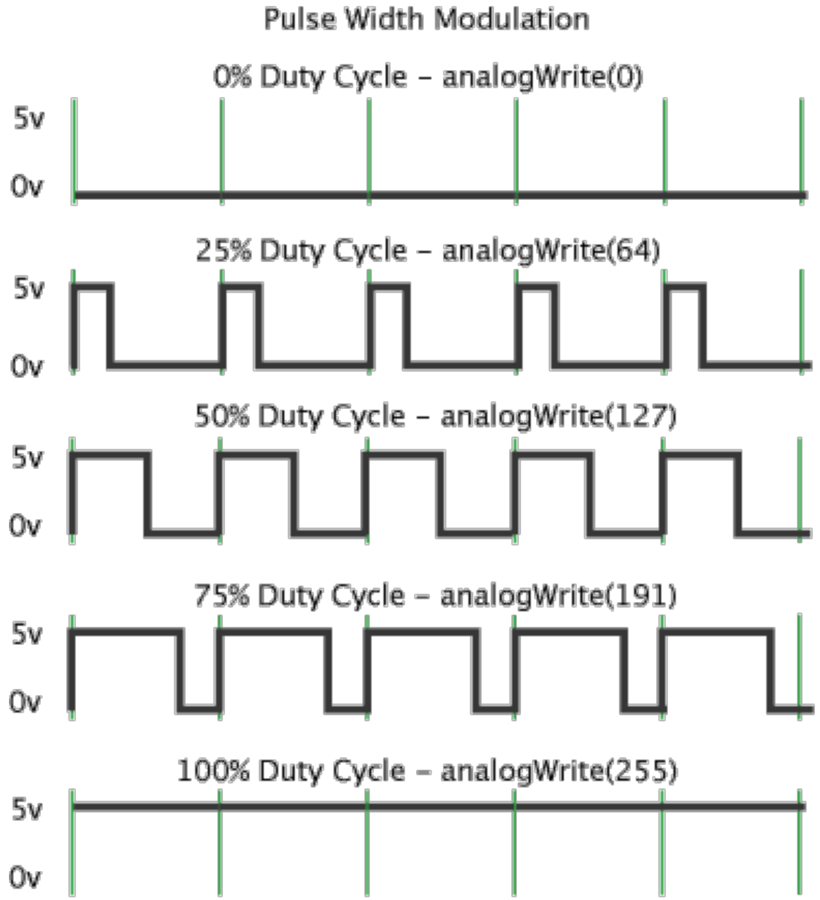
```
#include <Arduino.h>
void setup()
{
  pinMode(D9, OUTPUT); // Setup pin to be an output (this is the LED pin)
}

void loop()
{
  digitalWrite(D9, HIGH); // Turn the LED ON
  delay(500); // Leave it on for 500ms
  digitalWrite(D9, LOW); // Turn the LED OFF
  delay(500); // Leave it off for 500ms
}
```



Pulse Width Modulation (PWM)

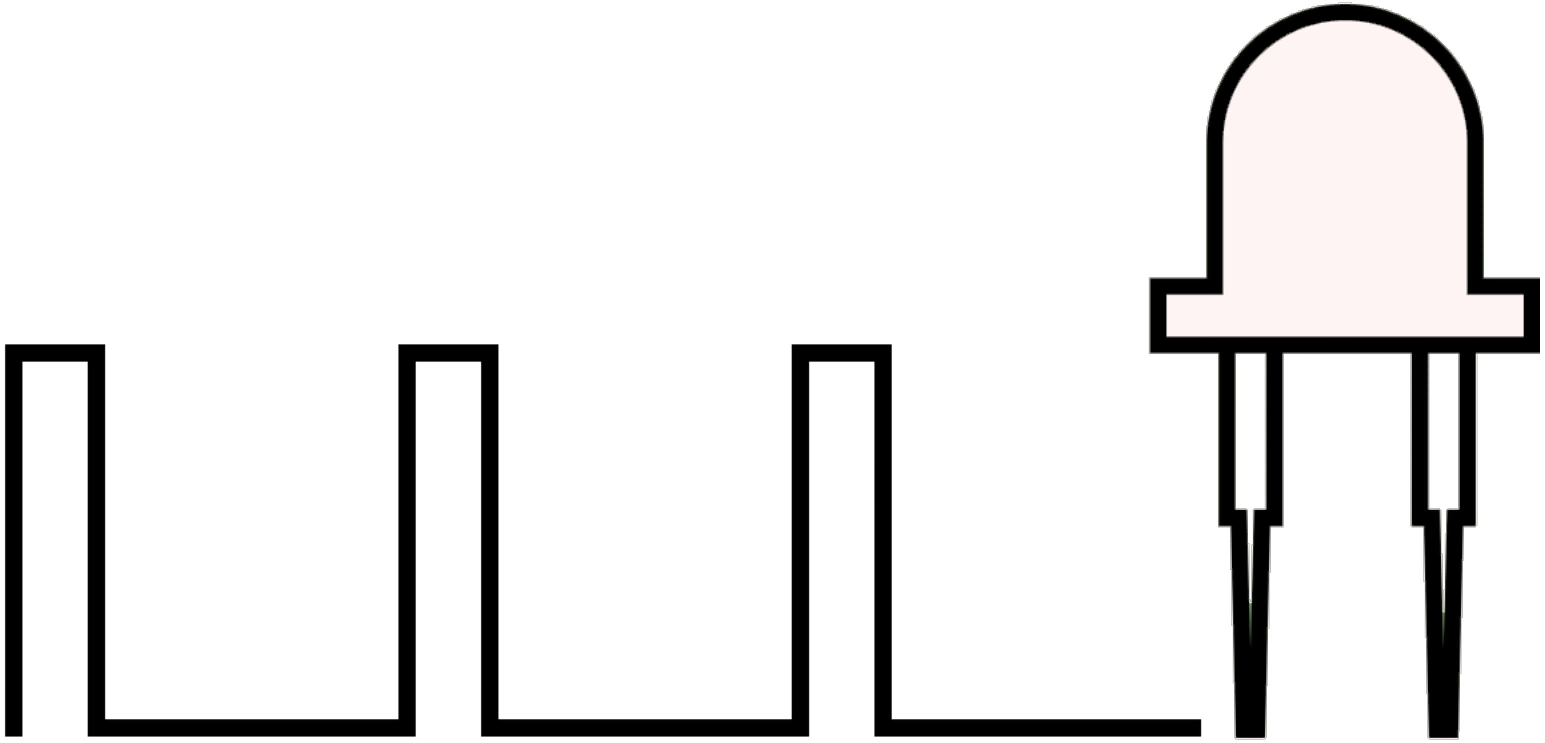
- In digital devices, hard to control the output intensity of signals
  - It is either 0 (low) or 1 (high)

- PWM – Uses a periodic digital signal with controlled pulse width


Pulse Width Modulation
0% Duty Cycle – analogWrite(0)
5v
0v
25% Duty Cycle – analogWrite(64)
5v
0v
50% Duty Cycle – analogWrite(127)
5v
0v
75% Duty Cycle – analogWrite(191)
5v
0v
100% Duty Cycle – analogWrite(255)
5v
0v

arduino.cc/en/tutorial/PWM

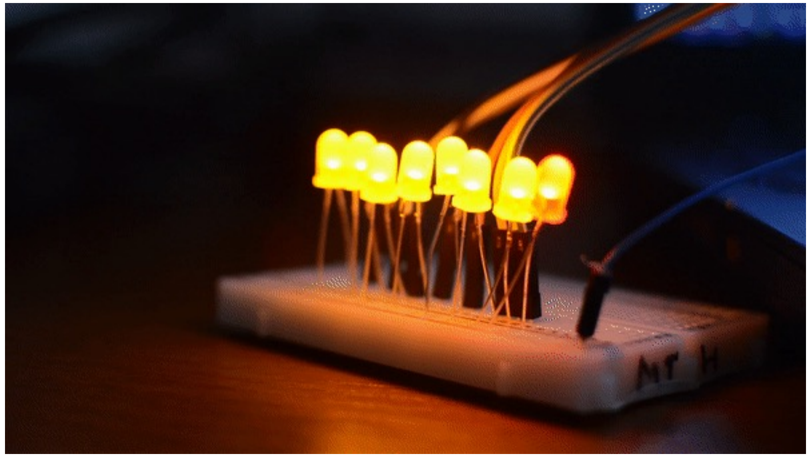
Pulse Width Modulation (PWM)

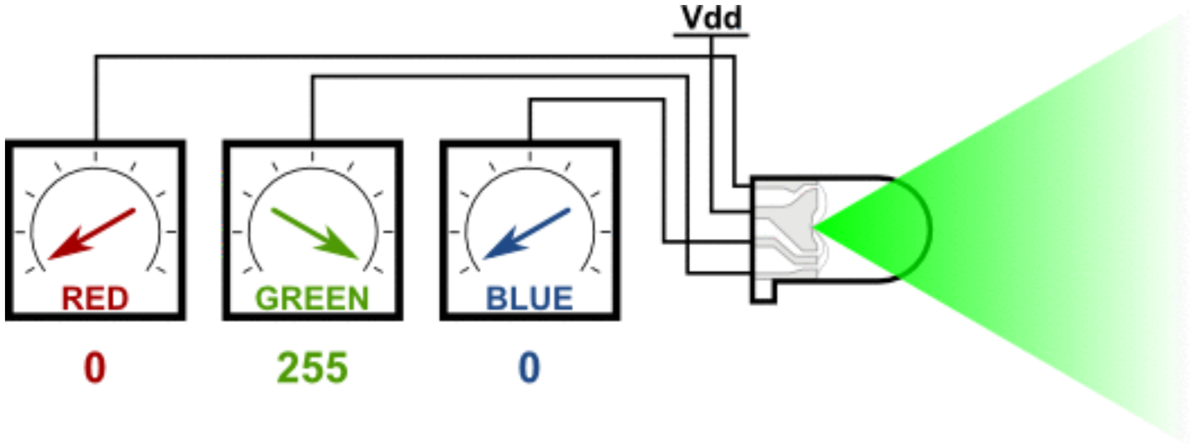


Leverages the **Persistence of vision**

www.electronicwings.com/arduino/pwm-in-arduino

Pulse Width Modulation (PWM)

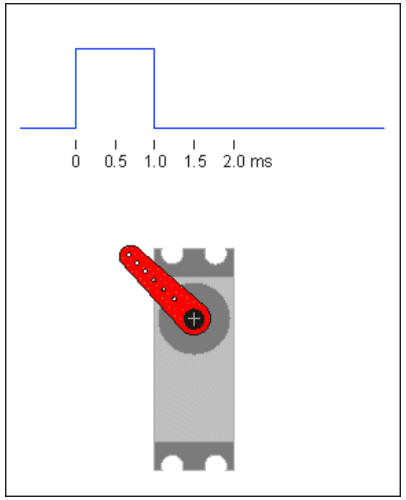


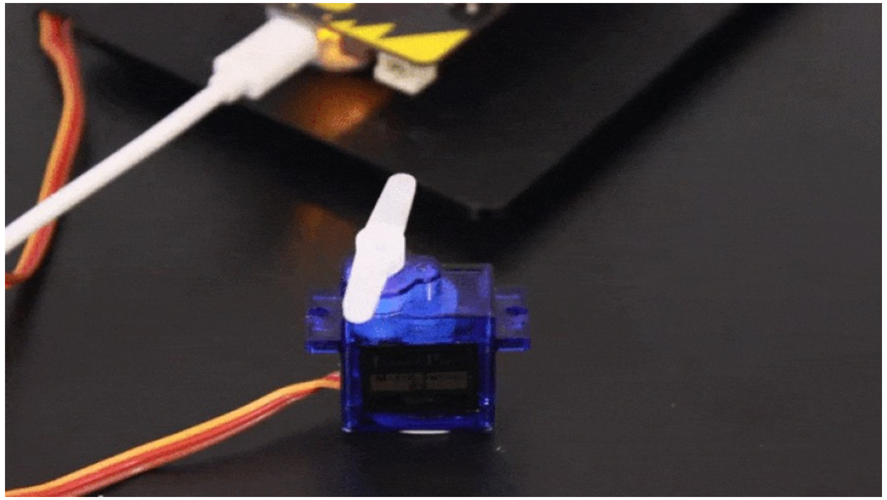


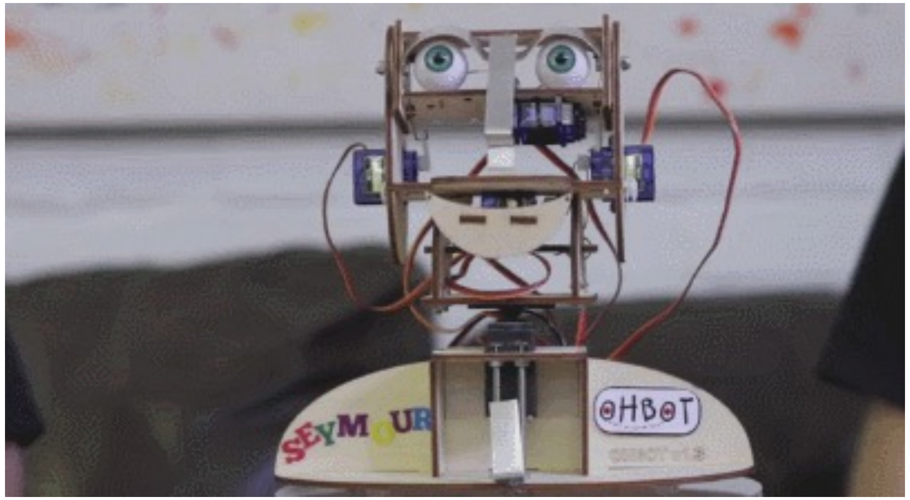
mtheelen.nl/pwm-pins-fading-leds/
giphy.com/gifs/led-NrzDtKdaXpNyo

Pulse Width Modulation (PWM)

- Also used as a communication protocol



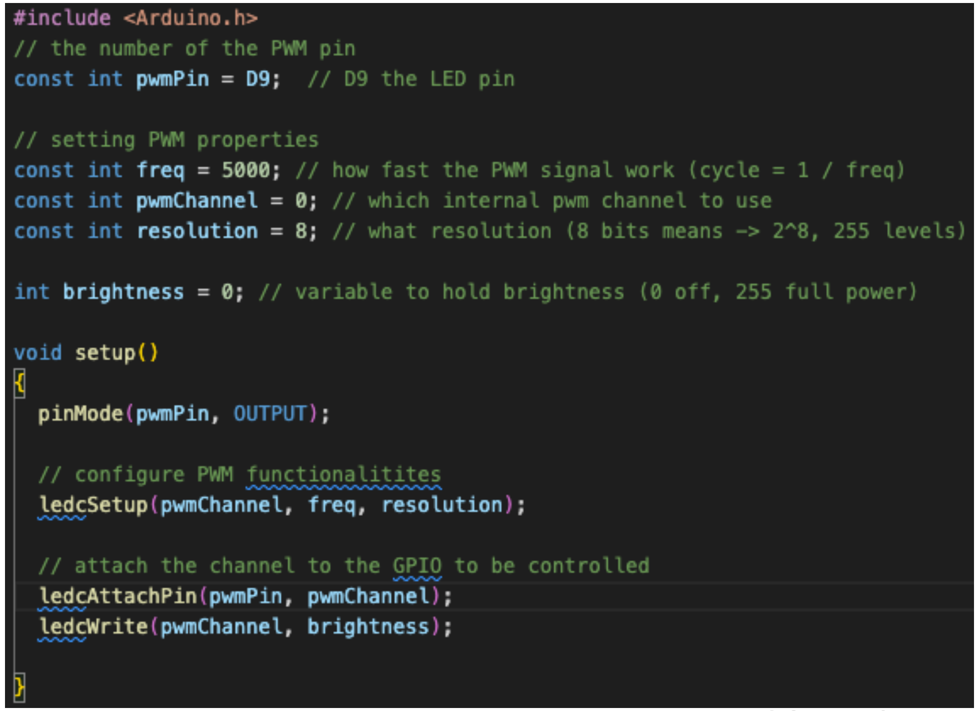




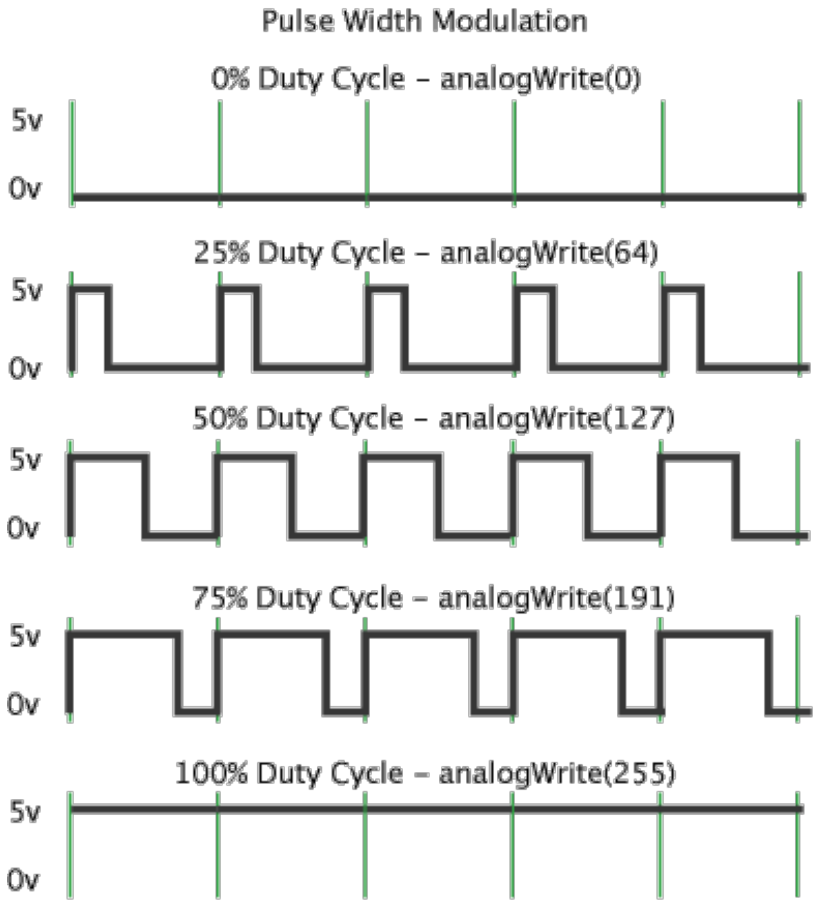
kroboblogs.wordpress.com
kitronik.co.uk/blog/using-bbc-microbit-control-servo
.roboticgizmos.com/ohbot2-programmable-robot-head

Pulse Width Modulation (PWM)

- Usually easy in Arduino
  - analogWrite(pin, value)
  - But with ESP (your µC) bit difficult



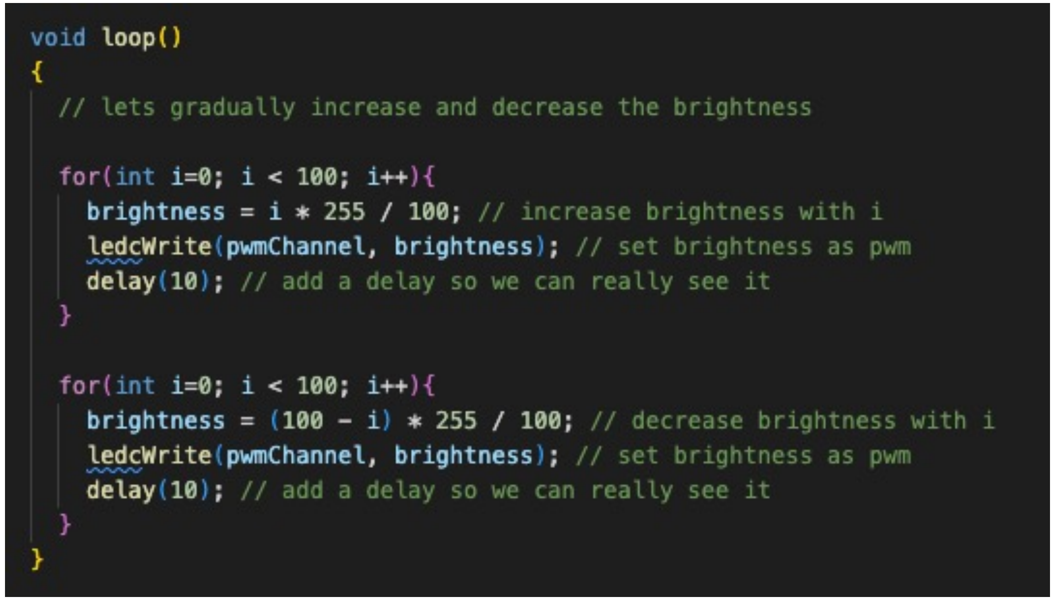
```
#include <Arduino.h>
// the number of the PWM pin
const int pwmPin = D9;  // D9 the LED pin

// setting PWM properties
const int freq = 5000; // how fast the PWM signal work (cycle = 1 / freq)
const int pwmChannel = 0; // which internal pwm channel to use
const int resolution = 8; // what resolution (8 bits means -> 2^8, 255 levels)

int brightness = 0; // variable to hold brightness (0 off, 255 full power)

void setup()
{
  pinMode(pwmPin, OUTPUT);

  // configure PWM functionalitites
  ledcSetup(pwmChannel, freq, resolution);

  // attach the channel to the GPIO to be controlled
  ledcAttachPin(pwmPin, pwmChannel);
  ledcWrite(pwmChannel, brightness);

}
```


Pulse Width Modulation
0% Duty Cycle – analogWrite(0)
5v
0v
25% Duty Cycle – analogWrite(64)
5v
0v
50% Duty Cycle – analogWrite(127)
5v
0v
75% Duty Cycle – analogWrite(191)
5v
0v
100% Duty Cycle – analogWrite(255)
5v
0v

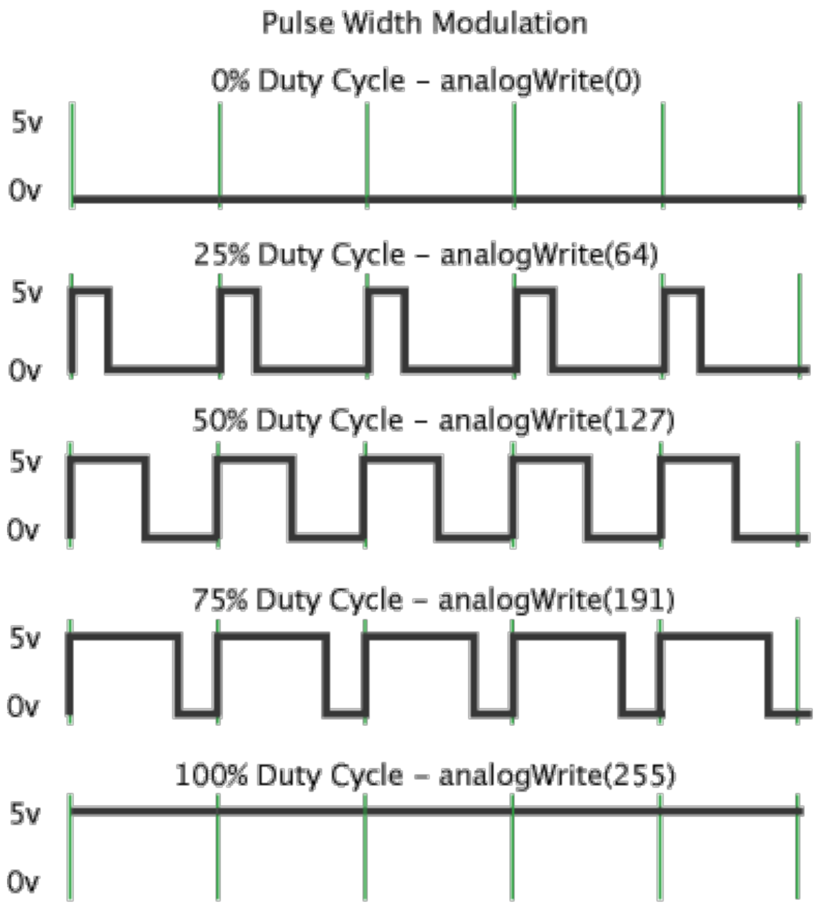
arduino.cc/en/tutorial/PWM

Pulse Width Modulation (PWM)

- Usually easy in Arduino
  - analogWrite(pin, value)
  - But with ESP (your µC) bit difficult



```
void loop()
{
  // lets gradually increase and decrease the brightness

  for(int i=0; i < 100; i++){
    brightness = i * 255 / 100; // increase brightness with i
    ledcWrite(pwmChannel, brightness); // set brightness as pwm
    delay(10); // add a delay so we can really see it
  }

  for(int i=0; i < 100; i++){
    brightness = (100 - i) * 255 / 100; // decrease brightness with i
    ledcWrite(pwmChannel, brightness); // set brightness as pwm
    delay(10); // add a delay so we can really see it
  }
}
```

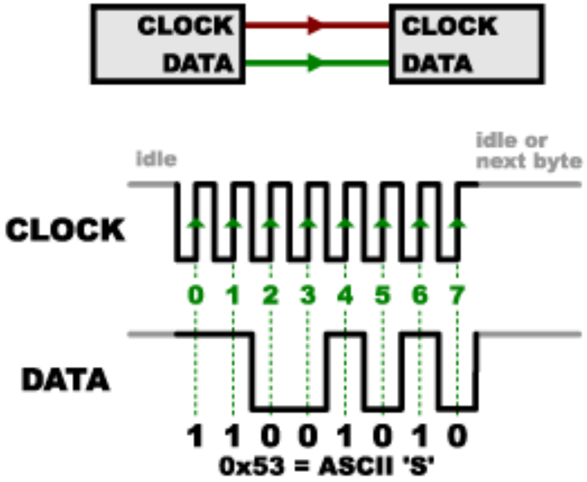


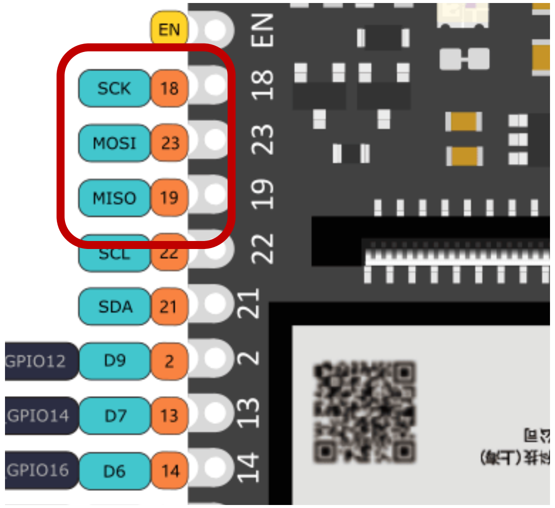
Pulse Width Modulation
0% Duty Cycle – analogWrite(0)
5v
0v
25% Duty Cycle – analogWrite(64)
5v
0v
50% Duty Cycle – analogWrite(127)
5v
0v
75% Duty Cycle – analogWrite(191)
5v
0v
100% Duty Cycle – analogWrite(255)
5v
0v

arduino.cc/en/tutorial/PWM

Serial Peripheral Interface (SPI)

- A Synchronous Communication Interface
  - Overcome the overheads of asynchronous (e.g. UART)
- Uses a clock (a pin send a clock signal)

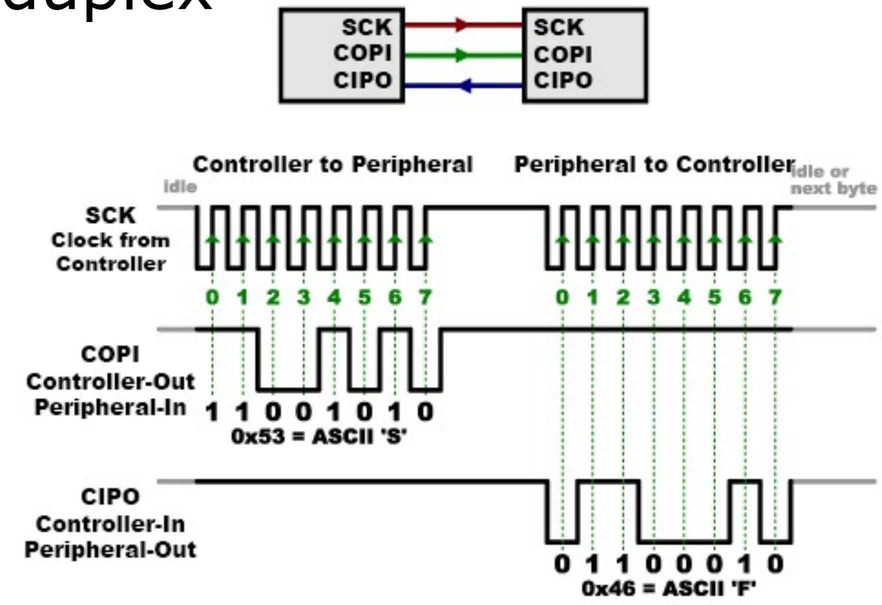


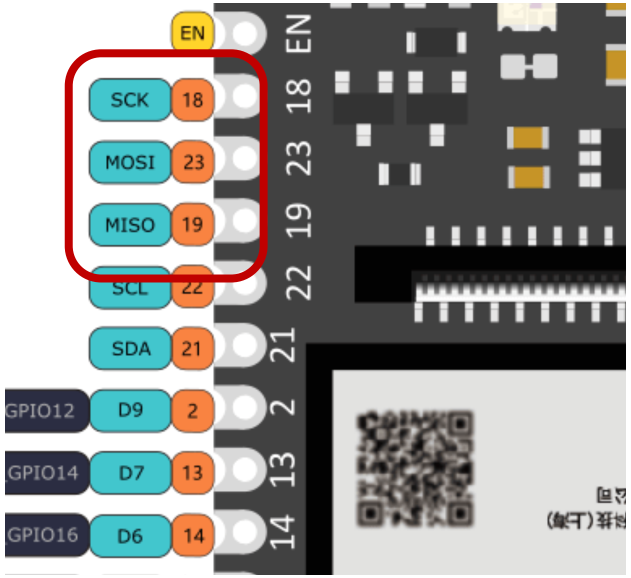


Adapted from: https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/all. (Highly recommend read)

Serial Peripheral Interface (SPI)

- Follows a **C**ontroller (**M**aster) and **P**eripheral (**S**lave) configuration
  - Full duplex

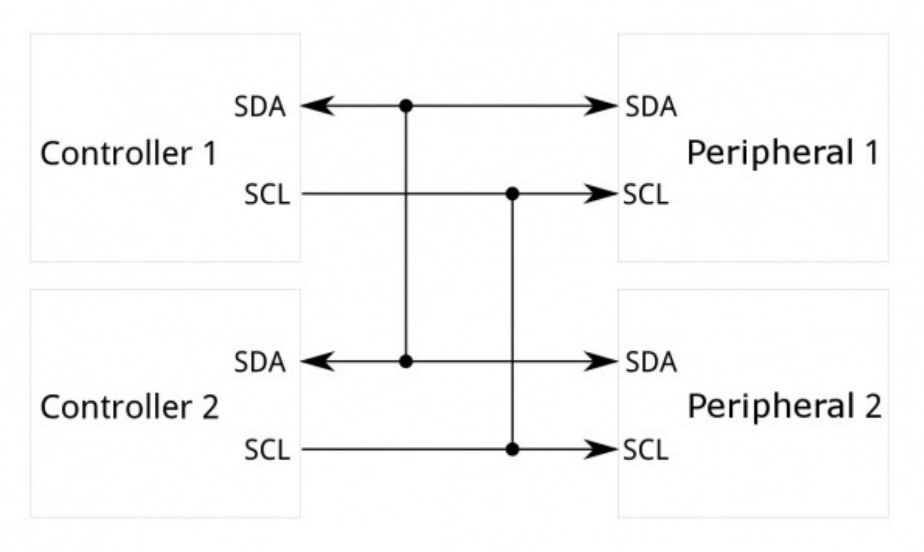


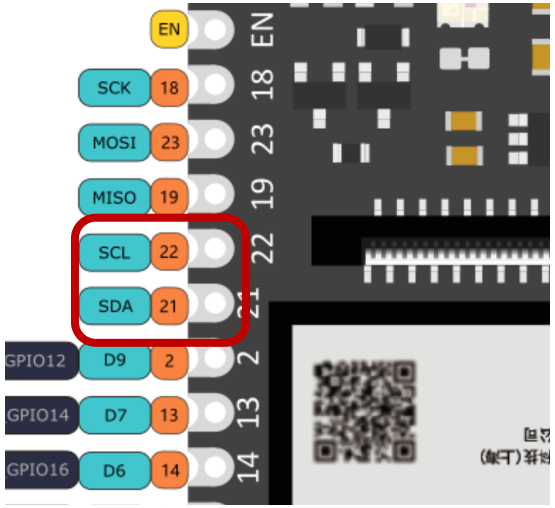


Adapted from: https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/all. (Highly recommend read)

I2C / I^{2}C (Inter-Integrated Circuit)

- Another Synchronous Communication Interface
- Uses an address per peripheral instead of **CS**
  - **Half duplex**

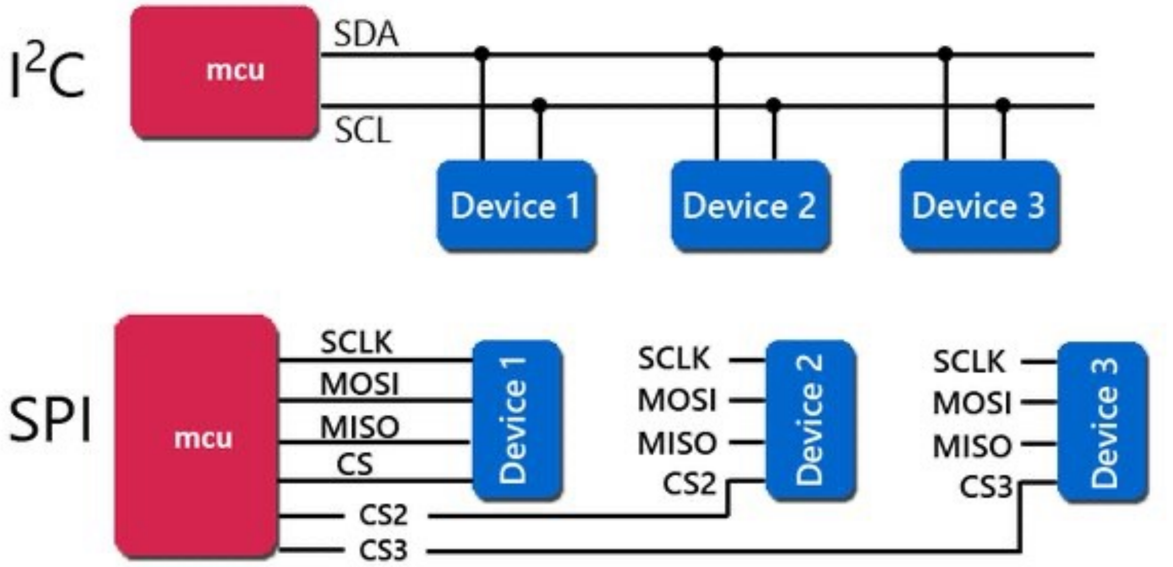




Adapted from: https://learn.sparkfun.com/tutorials/i2c . (Highly recommend read)

I2C / I^{2}C (Inter-Integrated Circuit)

- Complexity is reduced when there are many devices
  - But the data throughput get reduced
    - 400 kbit/s in fast modes (some other faster modes, but not common)
  - More devices means more noise



https://www.quora.com/

Summary

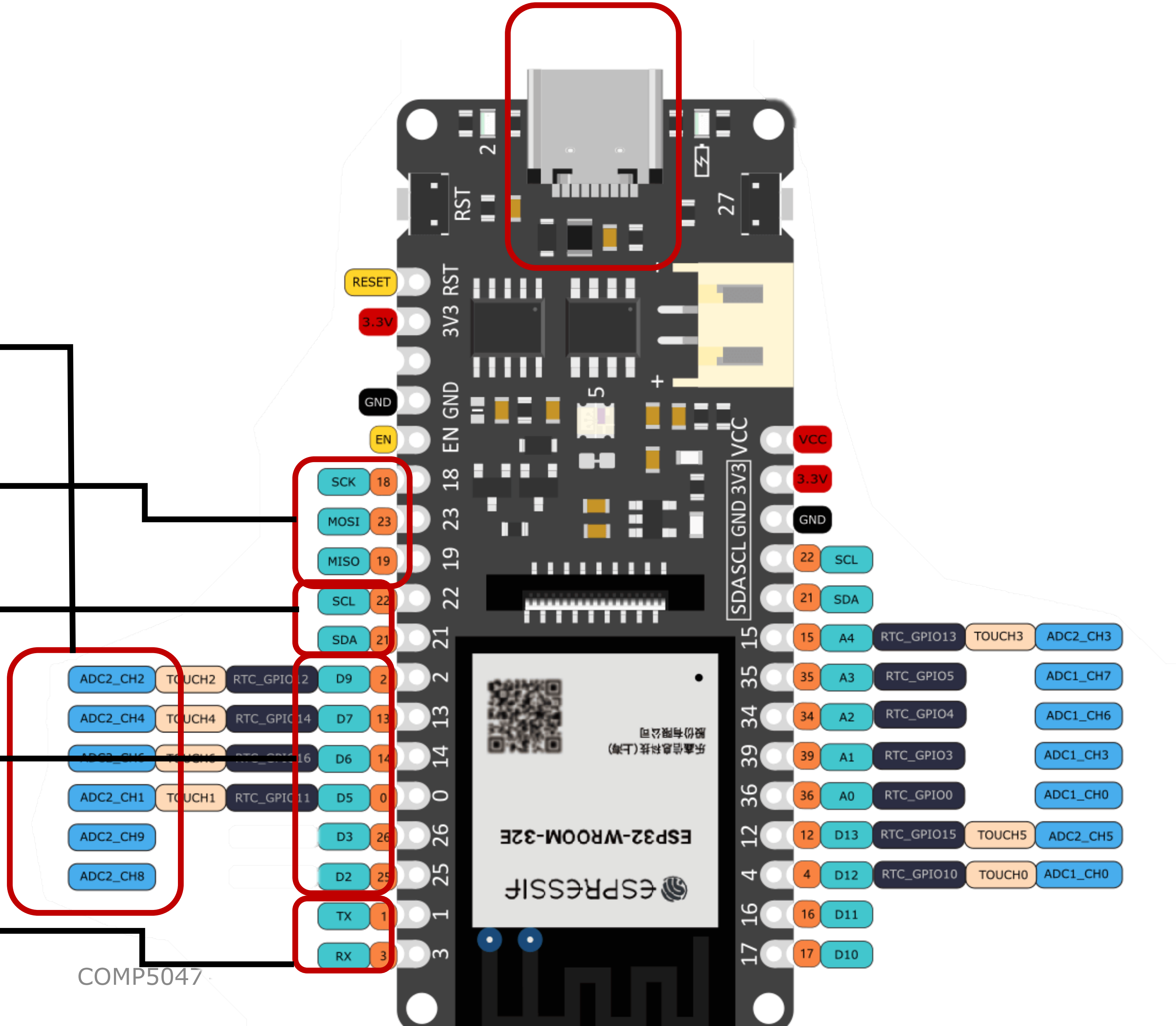
Analog to Digital Converter (**ADC**)

Serial Peripheral Interface (**SPI**)

I2C (Inter-Integrated Circuit, eye-squared-C)

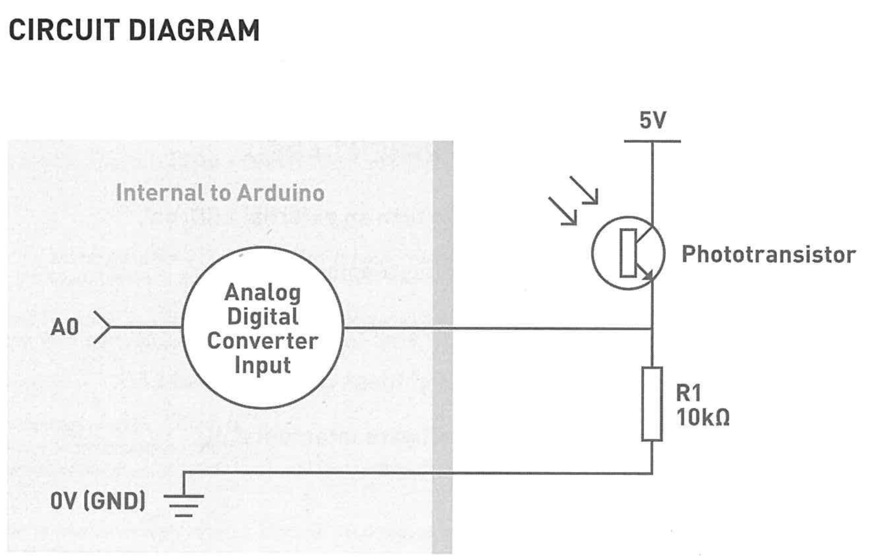
Digital IO / PWM

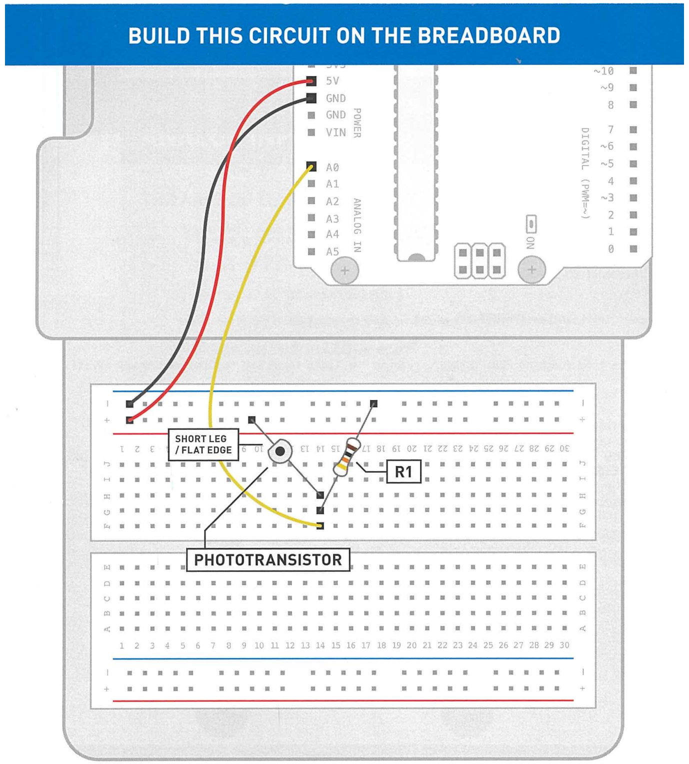
Universal Asynchronous Receiver/Transmitter (**UART**)/ **COM** port



What Are We Doing Today

Measure the ambient light in the room using the Phototransistor


CIRCUIT DIAGRAM
Internal to Arduino
Analog Digital Converter Input
A0
0V (GND)
5V
Phototransistor
R1
10kΩ


BUILD THIS CIRCUIT ON THE BREADBOARD
5V
GND
GND
VIN
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN
DIGITAL (PWM=~)
ON
SHORT LEG / FLAT EDGE
R1
PHOTOTRANSISTOR