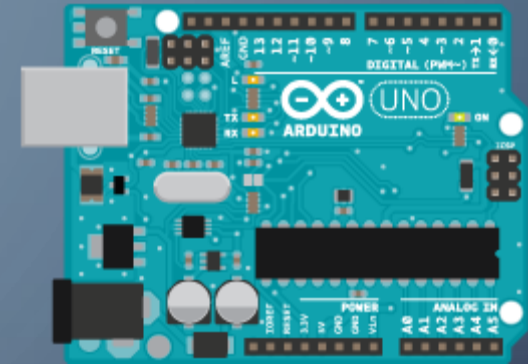


GETTING STARTED WITH ARDUINO

OPEN-SOURCE ELECTRONICS PROTOTYPING PLATFORM



By: Mayank Jain, Demonstrator, Programming Club, The University of Edinburgh | 2018-19

WHY ARDUINO?

- Control Hardware via Programming
- Simple and Intuitive Interface
- Open-Source – Hardware + Software
- Compatibility with most Electronic Modules in the market
- Affordable
- Low Power Consumption

WHAT IS ARDUINO?

- Microcontroller Based Prototyping Platform
- Programmable via Arduino IDE over USB
- Capable to be interfaced with:
 - Indicators
 - Actuators
 - Sensors
- Works in Real-Time – Application Specific Design

The background is a dark blue gradient. In the corners, there are white line-art illustrations of electronic circuit traces. These traces consist of straight lines of varying lengths and angles, connected by small circles, resembling a printed circuit board layout. The lines are more prominent in the corners and fade slightly towards the center.

REVISITING ELECTRONICS


LET'S BRUSH-UP SOME ELECTRONICS CONCEPTS

ELECTRONICS TERMS

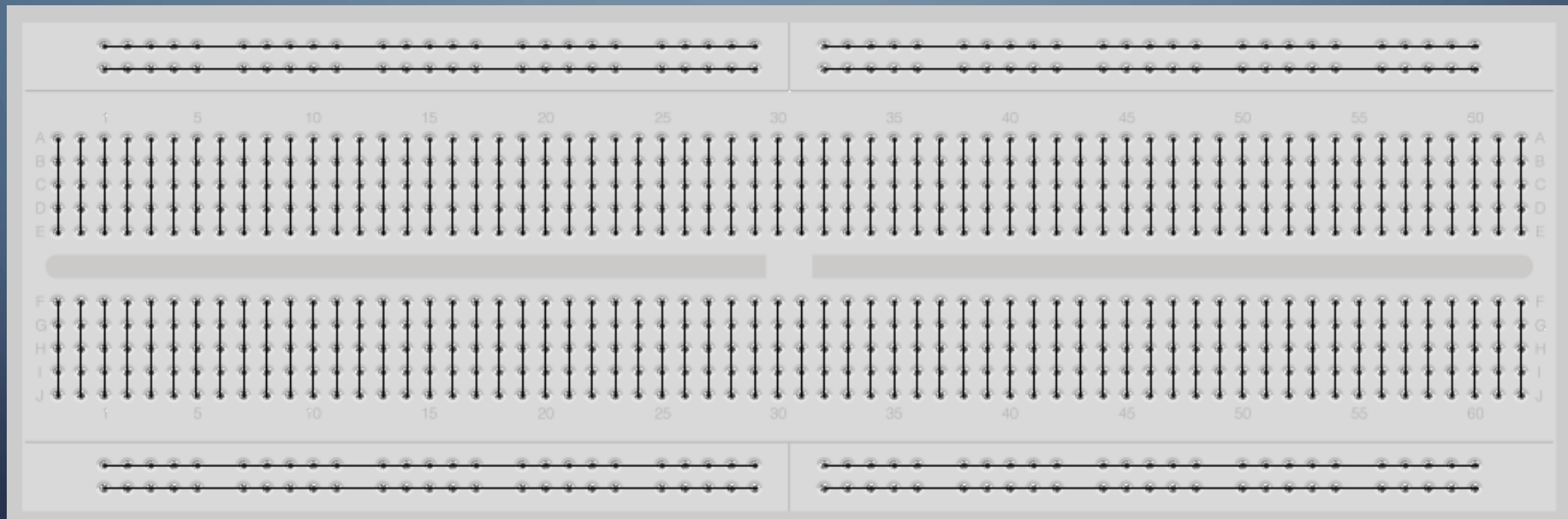
- Voltage (V) – Volts (V)
- Current (I) – Amperes (A)
- Resistance (R) – Ohms (Ω)
- Ohm's Law :: $V = IR$
- Alternating Current (AC)
- Direct Current (DC)



ELECTRONIC DEVICES

- Battery
 - Wire
 - Bulb
 - Diode/LED
 - Motor
 - Breadboard
- 
- 

BREADBOARD



LED



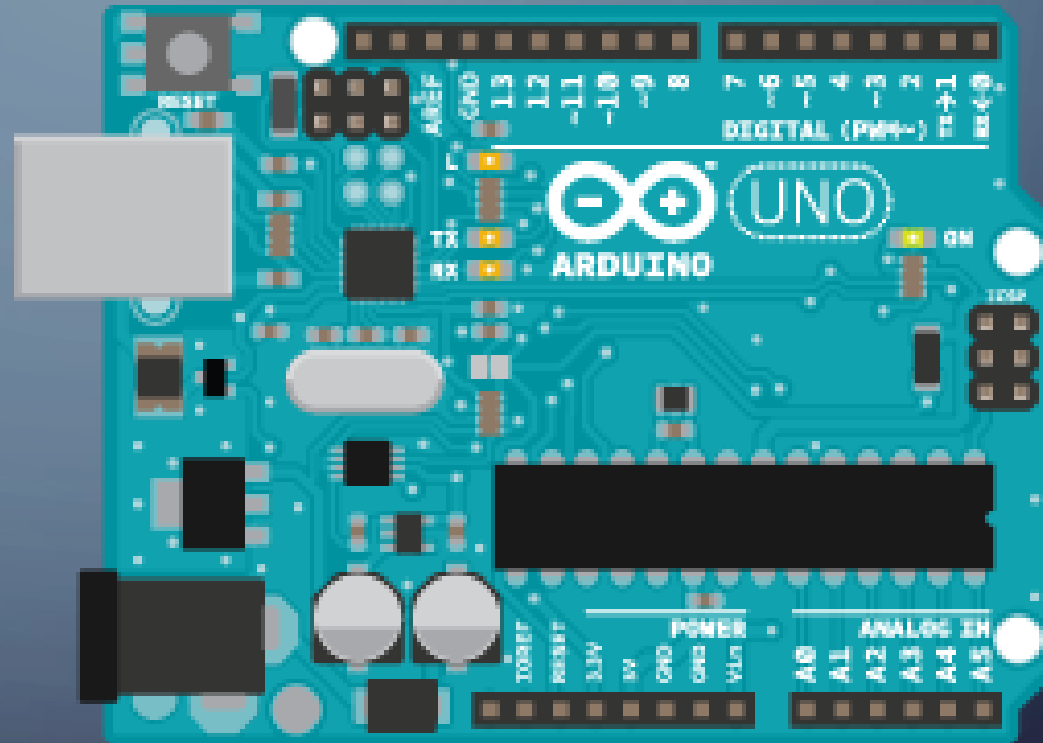
The background is a dark blue gradient. In the corners, there are white line-art illustrations of circuit boards. The top-left and bottom-left corners feature more complex, branching circuit patterns. The top-right and bottom-right corners have simpler, more linear circuit patterns.

RESUMING WITH ARDUINO

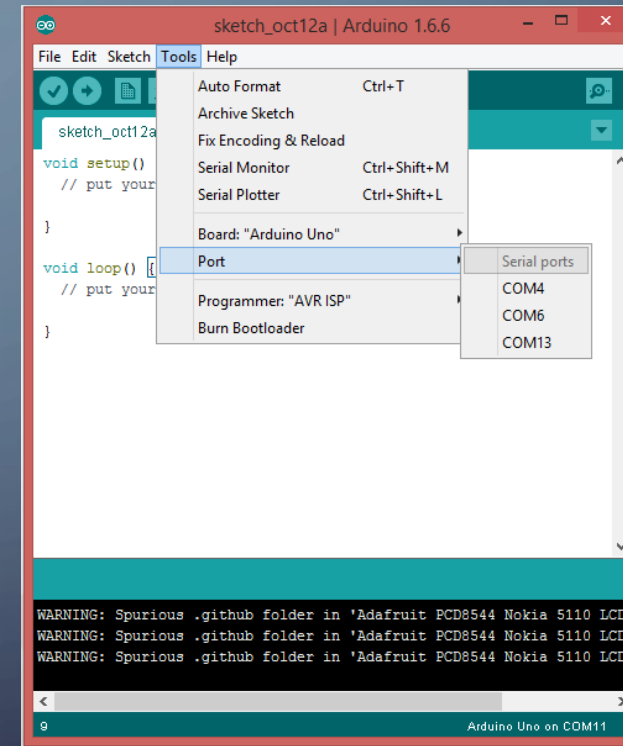
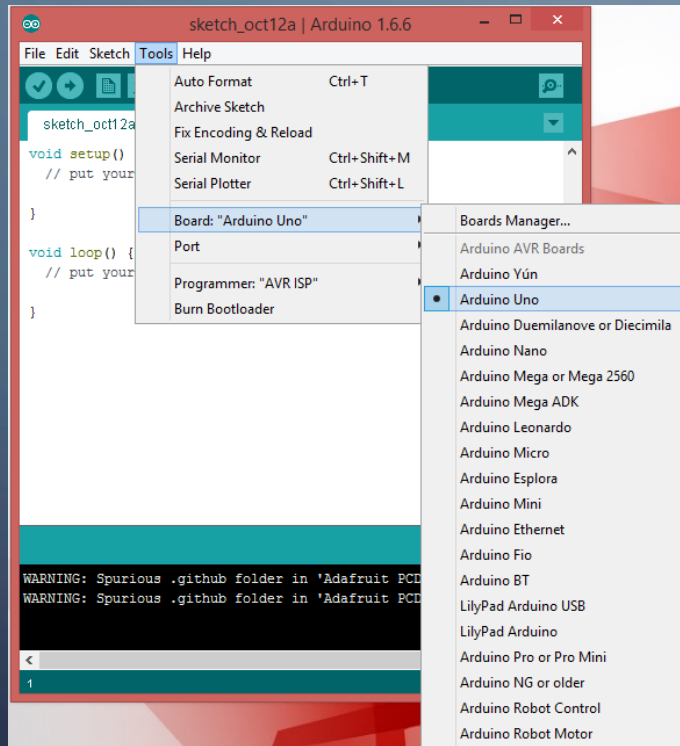
LET'S GET STARTED!

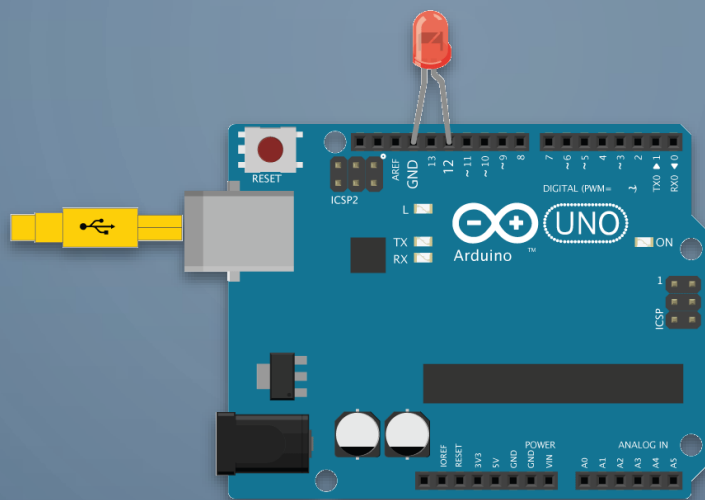
ARDUINO UNO

Microcontroller:	ATmega328P
Operating Voltage:	5V
Input Voltage:	7V – 12V
Digital I/O Pins:	14 (6 for PWM)
Analog Input Pins:	6
Flash Memory:	32KB (0.5KB Bootloader)
SRAM:	2KB
EEPROM:	1KB
Clock Speed:	16MHz
DC Current I/O Pin:	20mA
DC Current Power Pins:	50mA



SETTING UP IDE





PLAYING WITH LED

Make the circuit as shown above

#1 DEFINE LED PIN

```
int led = 12;  
void setup() {  
    pinMode(led, OUTPUT);  
}  
void loop(){  
}
```

#2 TURN LED ON

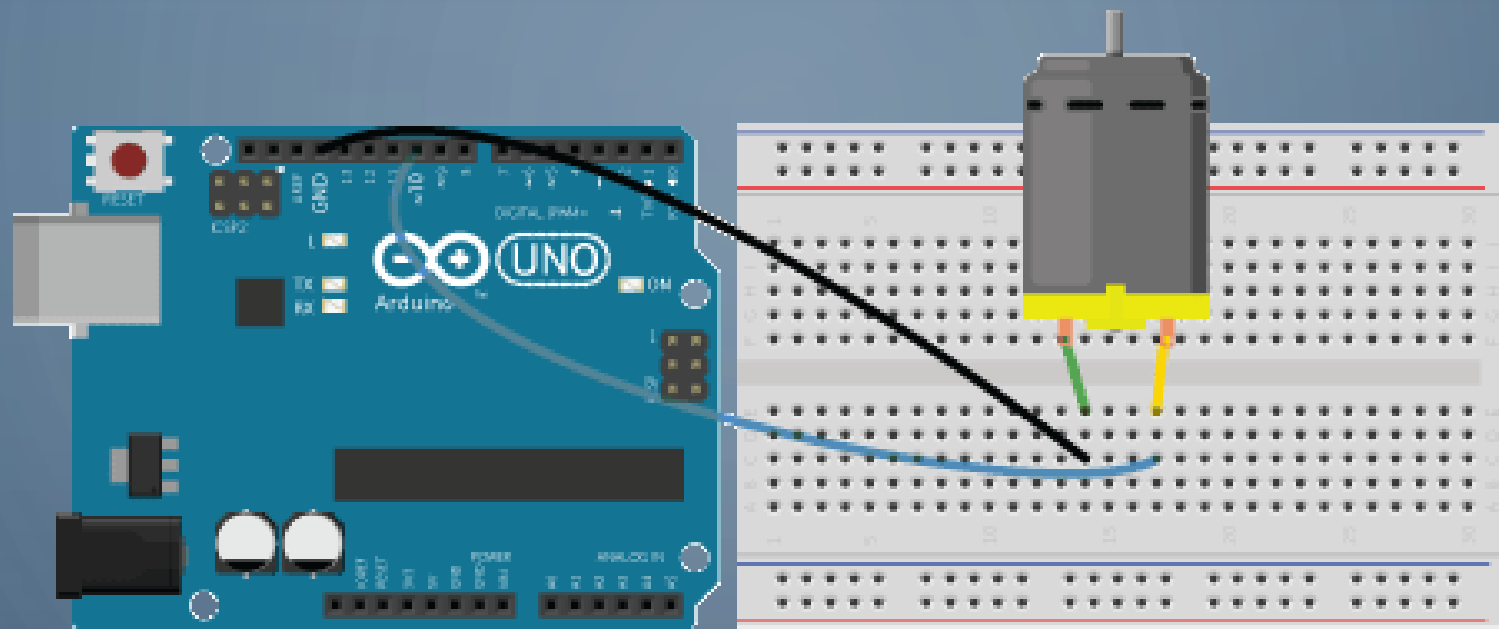
```
int led = 12;  
void setup() {  
    pinMode(led, OUTPUT);  
}  
void loop(){  
    digitalWrite(led, HIGH);  
}
```

#3 TURN LED ON AND OFF

```
int led = 12;
void setup() {
    pinMode(led, OUTPUT);
}
void loop(){
    digitalWrite(led, HIGH);
    digitalWrite(led, LOW);
}
```

#4 TURN LED ON AND OFF AND WAIT

```
int led = 12;
void setup() {
    pinMode(led, OUTPUT);
}
void loop(){
    digitalWrite(led, HIGH);
    delay(1000);
    digitalWrite(led, LOW);
    delay(1000);
}
```

PLAYING WITH MOTOR

Make the circuit as shown above

#1 TURN MOTOR ON AND OFF AND WAIT

```
int motor = 10;
void setup() {
    pinMode(motor, OUTPUT);
}
void loop(){
    digitalWrite(motor, HIGH);
    delay(1000);
    digitalWrite(motor, LOW);
    delay(1000);
}
```

#2 REVERSE DIRECTION OF MOTION



The background is a dark blue gradient. In the corners, there are white line-art illustrations of circuit traces and nodes. Top-left: several lines with circular nodes. Top-right: a few lines with circular nodes. Bottom-left: a cluster of lines with circular nodes. Bottom-right: a few lines with circular nodes.

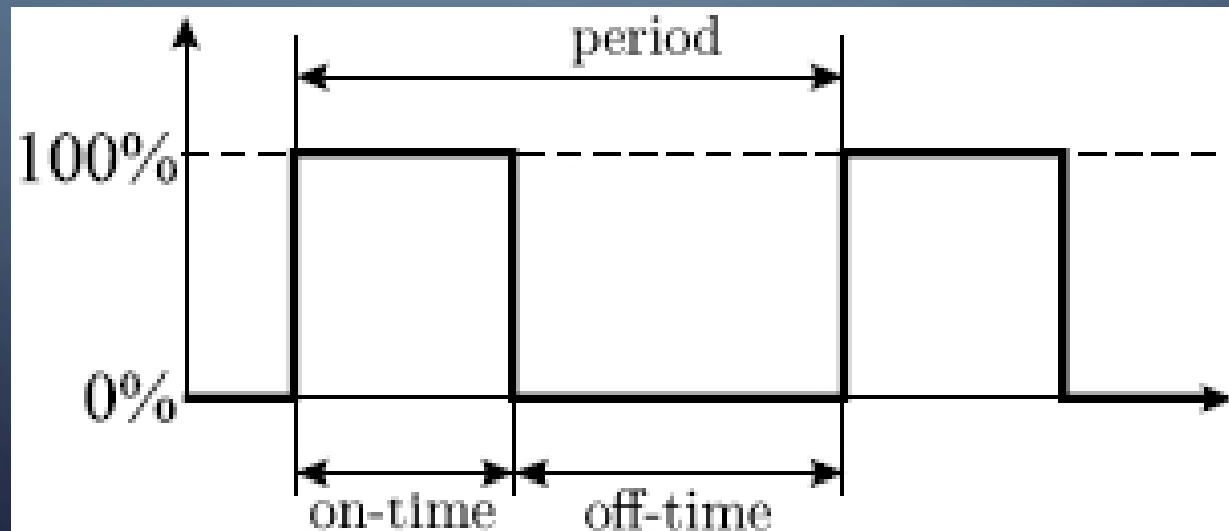
PULSE WIDTH MODULATION

ADJUSTING OUTPUT VOLTAGE

DEFINITIONS

- Duty Cycle: on-time / period
- V_{LOW} is often zero

$$V_{AVG} = DV_{HI} + (1 - D)V_{LOW}$$



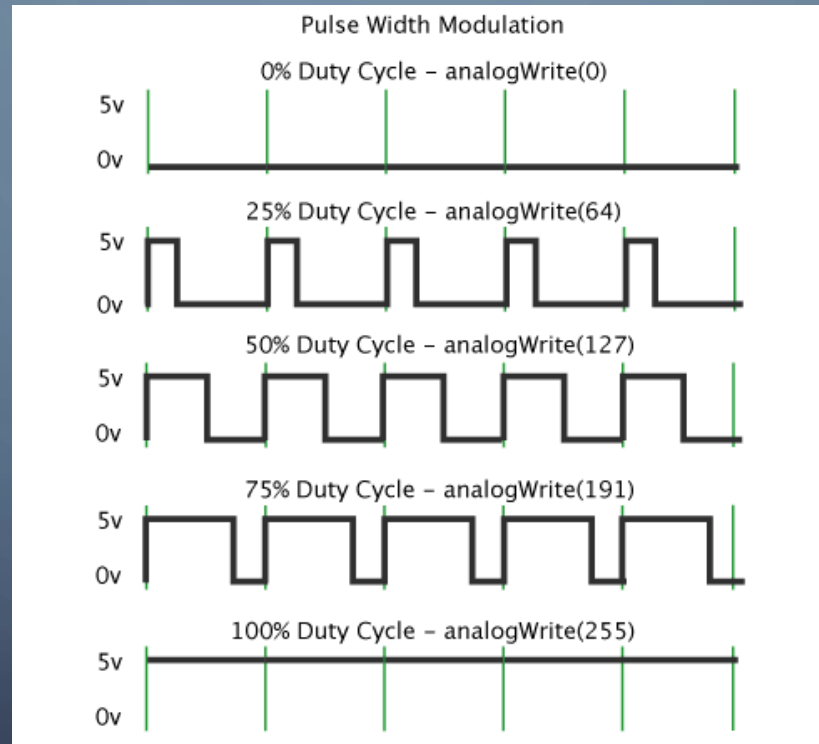
WHAT IS PWM?

- PWM – Pulse Width Modulation
- Output signal alternates between on and off within specified period
- Controls power received by a device
- The voltage seen by the load is directly proportional to the source voltage

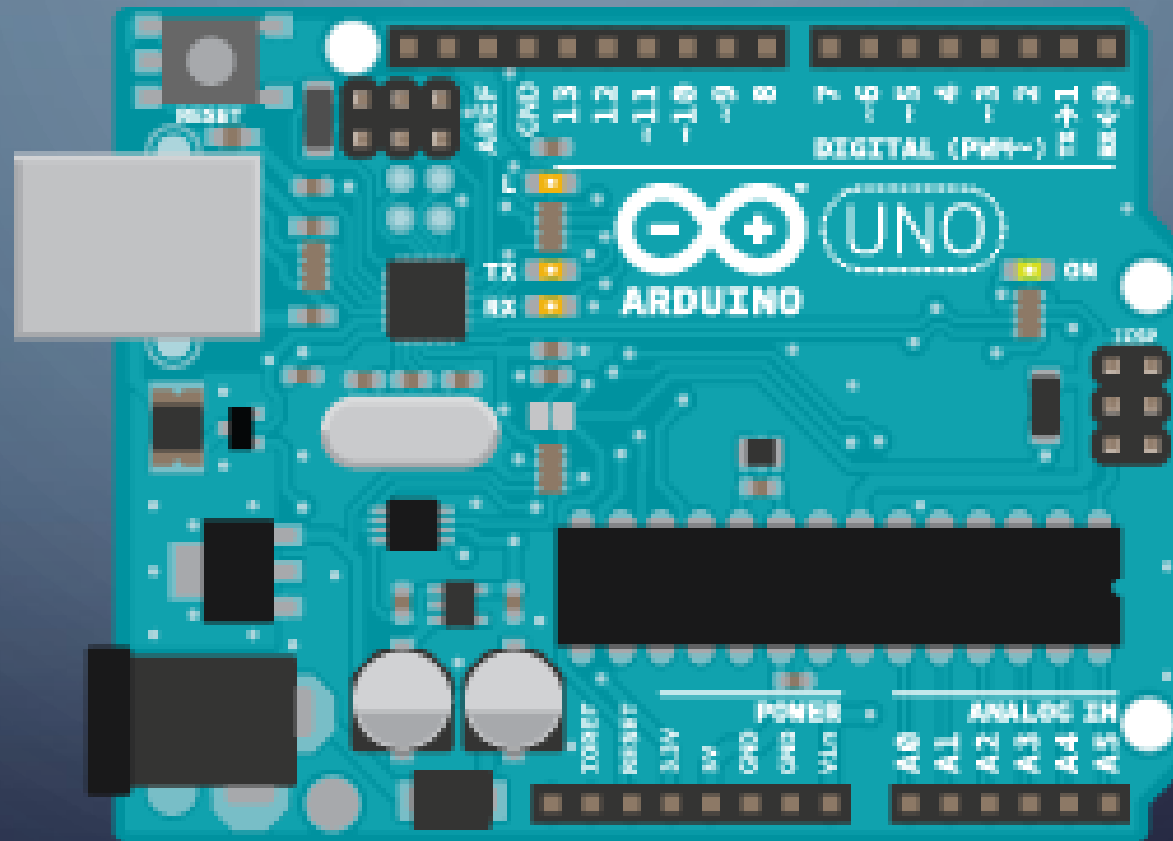
PWM - THEORY

- Technique for getting analog values from digital means
- Digital control is used to create a square wave, a signal switched between on and off.
- This on-off pattern can simulate voltages in between full on (5 Volts) and off (0 Volts) by changing the portion of the time the signal spends on versus the time that the signal spends off.
- The duration of "on time" is called the pulse width.
- To get varying analog values, you change, or modulate, that pulse width.

PWM - ARDUINO



ARDUINO UNO

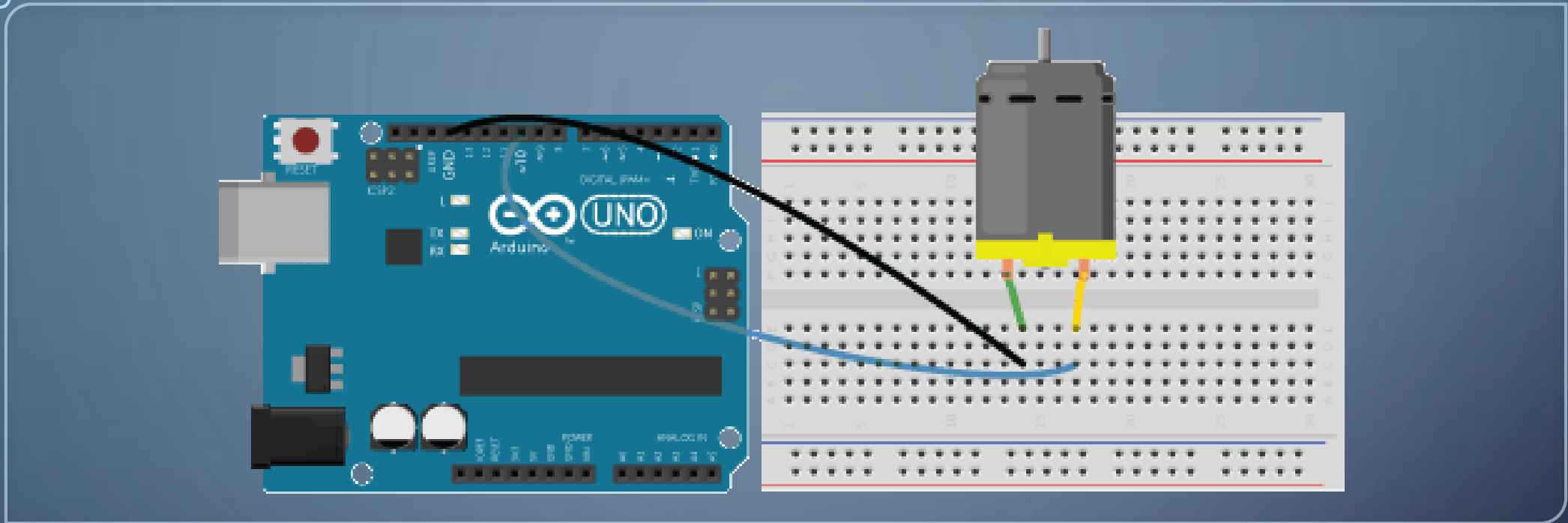


IMPLEMENTATION - UNO

- 6 PWM compatible pins (3, 5, 6, 9, 10, 11)
- Uno's PWM Frequency – 500 Hz (approx.)
 - Pin 5 and 6 have double this frequency
- Output value can vary from 0 V to 5 V
- Range can be divided in 256 parts

FUNCTION (MADE LIFE EASY)

- PWM implementation is implicitly done in Arduino
- `analogWrite(pin, value)`
 - *pin* : the pin to write to
 - *value* : the duty cycle: between 0 (always off) and 255 (always on)



PLAYING WITH MOTOR (SPEED CONTROL)

Make the circuit as shown above

#1 ADJUST THE SPEED OF MOTOR

```
int motor = 10;
void setup() {
    pinMode(motor, OUTPUT);
}
void loop(){
    analogWrite(motor, 200);
    delay(1000);
    analogWrite(motor, 100);
    delay(1000);
}
```

REFERENCES

- <https://www.arduino.cc/>
- <http://fritzing.org/home/>
- <https://www.arduino.cc/en/Tutorial/PWM>