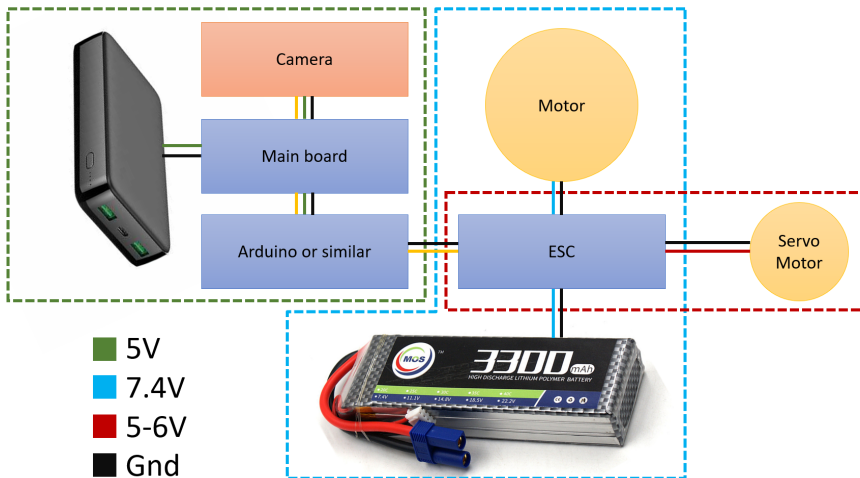


06 - Hardware, communication and protocols

Maxime Ellerbach

May 2023

Car Hardware Overview



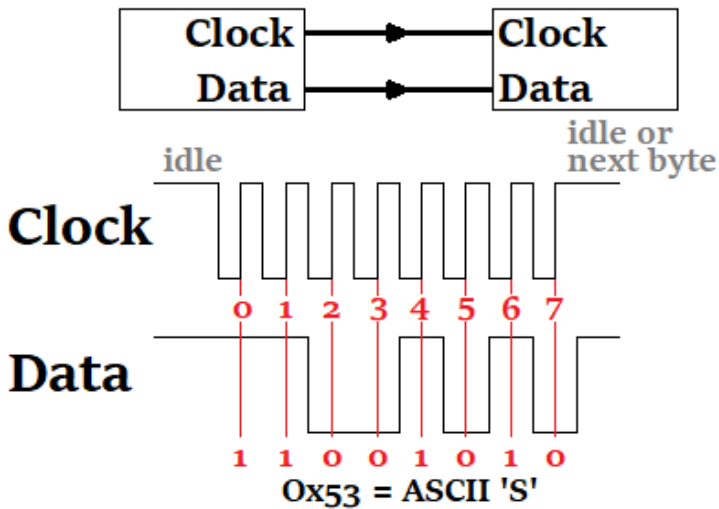
- Parallel Buses

- Multiple bits transmitted at the same time through separate lines
- Requires a lot of wiring e.g. 32 bits at same time = 32 wires
- ATA, SCSI, PCI, ...

- Serial Buses

- Transmits data bit by bit sequentially
- Slower transmission rates
- Less wiring
- Telegraph, USB, UART, I2C, CAN, ...

Synchronous Serial Bus



Unicast and Multicast

- Unicast

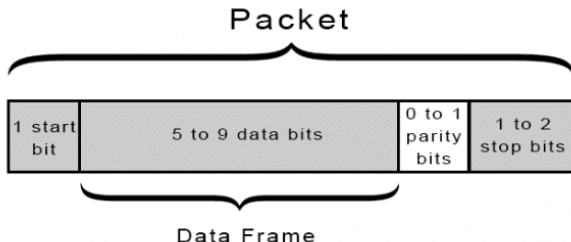
- Single sender & receiver
- point to point
- Limited scalability

- Multicast

- Single sender & multiple receiver
- One-to-many or many-to-many
- Efficient when having lots of receiver

UART - Universal Asynchronous Receiver/Transmitter

- **Asynchronous:** no shared clock
- **Serial:** single data line (RX / TX)
- **Half/Full Duplex:** can receive and send data simultaneously



<https://www.circuitbasics.com/>

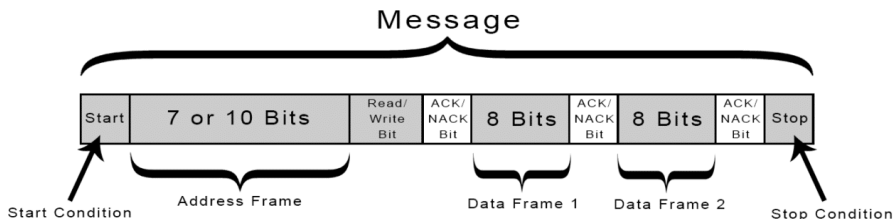
I2C - Inter-Integrated Circuit

- **Two communication lines**

- Serial Data Line (SDA)
- Serial Clock Line (SCL)

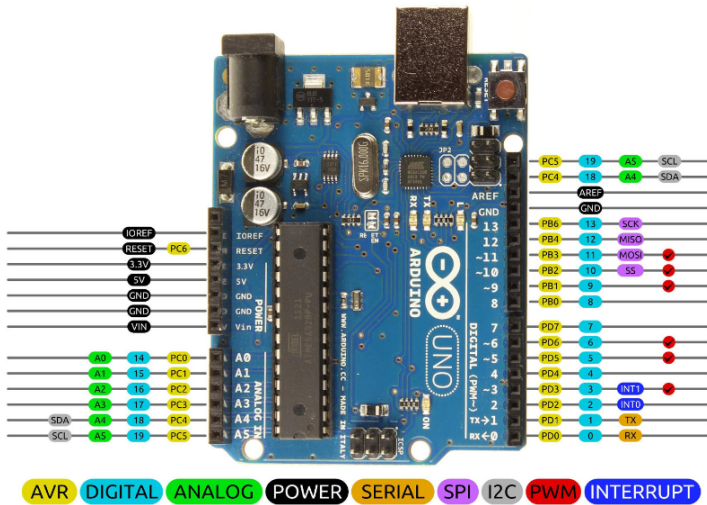
- **Master-Slave architecture**

- Supports multiple Master
- 7 bit addressing



<https://www.circuitbasics.com/>

Arduino Pinout



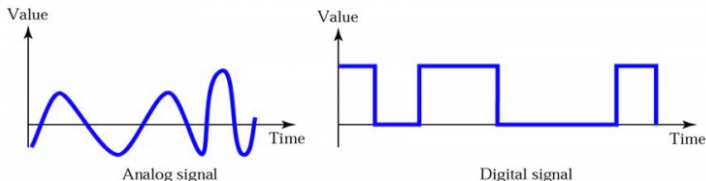
Digital and Analog pins

Digital Pins:

- Used to transmit and receive binary signals,
- Two voltage levels: HIGH (logic 1) or LOW (logic 0).

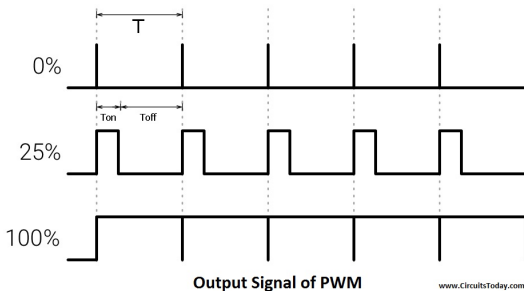
Analog Pins:

- Used to measure continuous or analog signals, such as voltages or sensor readings
- Convert analog signals into digital values
- Often labeled with an "A" followed by a number (e.g., A0, A1, A2)



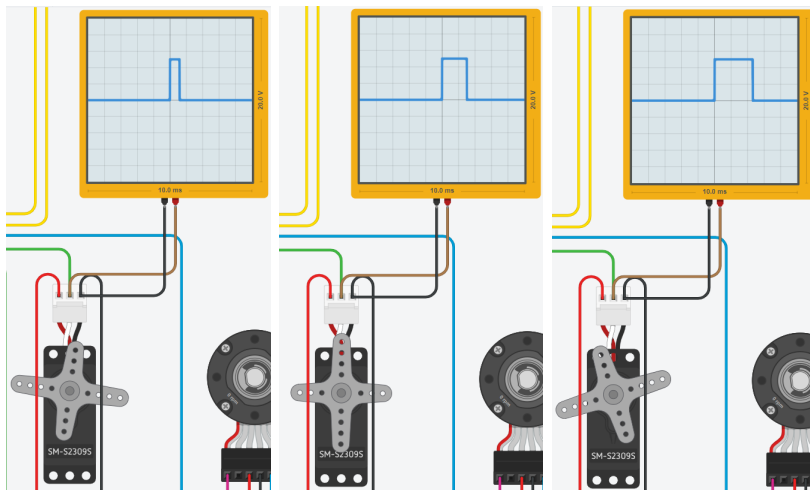
PWM (Pulse Width Modulation) Signal

- Encode analog information in a digital signal by varying the width of the pulses
- The output analog value is the ratio between the time the signal is HIGH (=1) and the period T
- Useful in our case to drive the servo and motor easily
- Only certain Digital pins support PWM, refer to the pinout of the micro-controller



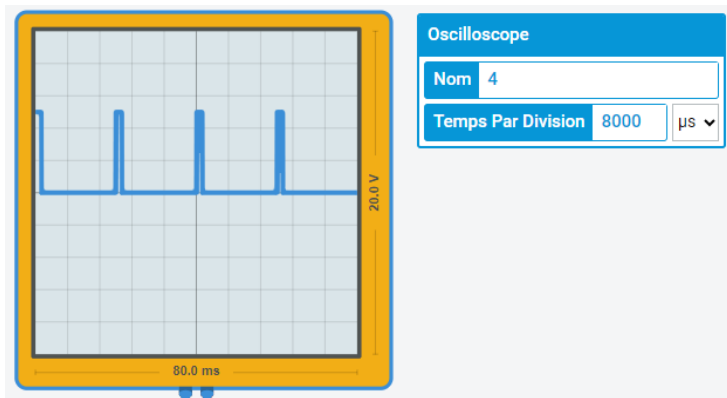
PWM (Pulse Width Modulation) Signal

left, center, right



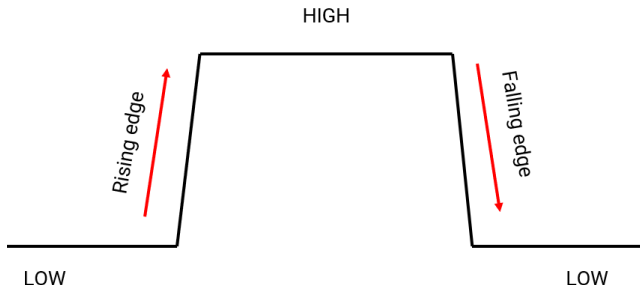
PWM (Pulse Width Modulation) Signal

- Here we observe 4 periods, so $T = 80\text{ms} / 4 = 20\text{ms}$
- Full left = 0.5ms HIGH time (500 microseconds). duty of 2.5%
- Full right = 2.5ms HIGH time (2500 microseconds) duty of 12.5%

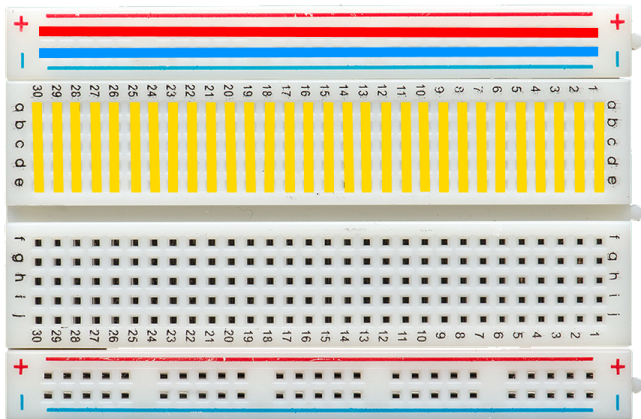


Interrupt pins

- Digital pin 2 and 3 on arduino Uno
- Trigger an interrupt request when specific event occurs.
 - **LOW** - triggers INT when pin is LOW
 - **RISING** - triggers INT when pin goes from LOW to HIGH
 - **FALLING** - triggers INT when pin goes from HIGH to LOW
 - **CHANGE** - triggers INT when pin changes state.



Breadboard wiring



Practice 05 - Tinkercad

05 ref - hardware serial

Toutes les modifications ont été enregistrées

Durée du simulateur: 00:01:44

Code Arrêter la simulation Envoyer ve...

1 (Arduino Uno R3)

```
1 // Servo motor control using PWM
2
3 // Pin definitions
4 #define SERVO_PIN 9
5 #define THROTTLE_PIN 10
6 #define STEERING_PIN 11
7
8 // Servo motor parameters
9 #define SERVO_MAX 180
10 #define SERVO_MIN 0
11
12 // Throttle and steering parameters
13 #define THROTTLE_MAX 255
14 #define THROTTLE_MIN 0
15 #define STEERING_MAX 180
16 #define STEERING_MIN 0
17
18 // Variables
19 int throttle = 0;
20 int steering = 0;
21 float decoded_steering = 0;
22
23 // Functions
24 void setup() {
25   pinMode(SERVO_PIN, OUTPUT);
26   pinMode(THROTTLE_PIN, OUTPUT);
27   pinMode(STEERING_PIN, OUTPUT);
28 }
29
30 void loop() {
31   // Read throttle and steering values
32   int throttle_val = analogRead(A0);
33   int steering_val = digitalRead(D0);
34
35   // Convert throttle to PWM
36   int pwm_throttle = map(throttle_val, 0, 1023, 0, 255);
37   analogWrite(THROTTLE_PIN, pwm_throttle);
38
39   // Convert steering to PWM
40   int pwm_steering = map(steering_val, 0, 1, 0, 180);
41   analogWrite(STEERING_PIN, pwm_steering);
42
43   // Delay
44   delay(100);
45 }
46
47 // Servo motor control using PWM
48
49 // Pin definitions
50 #define SERVO_PIN 9
51 #define THROTTLE_PIN 10
52 #define STEERING_PIN 11
53
54 // Servo motor parameters
55 #define SERVO_MAX 180
56 #define SERVO_MIN 0
57
58 // Throttle and steering parameters
59 #define THROTTLE_MAX 255
60 #define THROTTLE_MIN 0
61 #define STEERING_MAX 180
62 #define STEERING_MIN 0
63
64 // Variables
65 int throttle = 0;
66 int steering = 0;
67 float decoded_steering = 0;
68
69 // Functions
70 void setup() {
71   pinMode(SERVO_PIN, OUTPUT);
72   pinMode(THROTTLE_PIN, OUTPUT);
73   pinMode(STEERING_PIN, OUTPUT);
74 }
75
76 void loop() {
77   // Read throttle and steering values
78   int throttle_val = analogRead(A0);
79   int steering_val = digitalRead(D0);
80
81   // Convert throttle to PWM
82   int pwm_throttle = map(throttle_val, 0, 1023, 0, 255);
83   analogWrite(THROTTLE_PIN, pwm_throttle);
84
85   // Convert steering to PWM
86   int pwm_steering = map(steering_val, 0, 1, 0, 180);
87   analogWrite(STEERING_PIN, pwm_steering);
88
89   // Delay
90   delay(100);
91 }
92
93 void changeSteering()
94 {
95   // cast byte to float
96   float decoded_steering = buffData[0];
97   // scale steering to match servo specs
98   int steering = SERVO_MAX - decoded_steering / 255 * (SERVO_MAX - SERVO_MIN);
99   // apply steering
100   servoSteering.writeMicroseconds(steering);
101 }
102
103 void changeThrottle()
104 {
105   // cast byte to int
106   int throttle = buffData[1];
107   pwmToBridge(throttle);
108 }
109
110 // this function will be called on state change of SENSOR_PIN
111 void signalChange()
112 {
113   last_interrupt_time = micros();
114   if (digitalRead(SENSOR_PIN) == HIGH)
115   {
116     timer_start = last_interrupt_time;
```

11.7 V
445 mA

1 (Arduino Uno R3)

Moniteur série

4
2
0

Envoyer Effacer