IEEE Arduino Workshop series part II

The basics

Oscar Tesniere

McGill University

October 3, 2025

https://tinyurl.com/278v225m

Please make sure to download the presentation AND the exercise starter code Go to `Arduino workshop F25 > exercises` for exercices

You can find the slideshow at

`Arduino workshop F25 > part2.pdf`

Recap from Part I

- The arduino syntax and C++ programing language
- Pushbuttons, LEDs, Serial monitor

Today

Interrupts

libraries

Millis()

Morse example

Voltmeter

Part I QA

What is GND ?

What is the difference between uploading and compiling ?

What is the difference between analog and digital signals ?

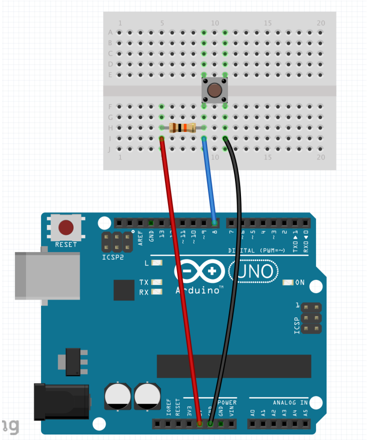
Does it make a difference to put the resistor before or after a LED

Wirin (breadboard) diagram vs circuit diagram

Arduino-specific syntax

- `pinMode(pin,mode)`: Configure a pin as INPUT or OUTPUT
- `digitalWrite(pin,value)`: Set a digital pin HIGH or LOW
- `digitalRead(pin)`: Read the state of a digital pin (HIGH/LOW)
- `analogRead(pin)`: Read an analog value (0–1023) from a pin
- `Serial` class:
  - `Serial.begin(baudRate)`: Initialize serial communication
  - `Serial.print()`, `Serial.println()`: Send data over serial

Let's get back to our pushbutton exercise



☞ Connect the following wiring diagram. You'll need a 2.2k or 2.7k resistor and an additional jumper wire. Upload the `fixed`$_b$*utton.ino*|*code*.

- **Does the value of pressCount change on its own now?**
  - `1x 2.2k or 2.7k resistor`
  - `1x pushbutton`
  - `3x jumper wires`

Why some configurations didn't work

- The digital pin is left floating and oscillates to LOW (triggering increment)
- The pushbutton bounces and generates parasite signal which increments the counter more than once / press

The pushbutton debounce problem

Solutions available

→ Capacitor

The pushbutton debounce problem

Solutions available

- Capacitor

→ Pull up resistor

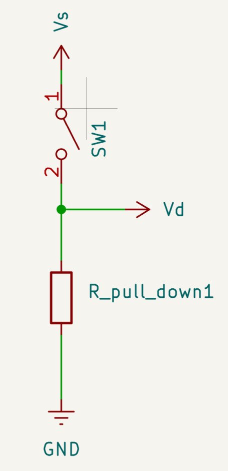
The pushbutton debounce problem

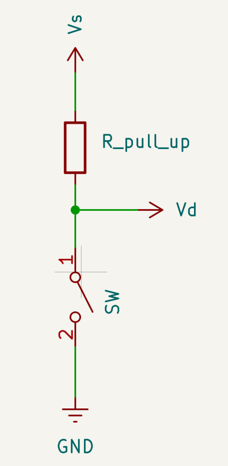
Solutions available

- Capacitor
- Pull up resistor

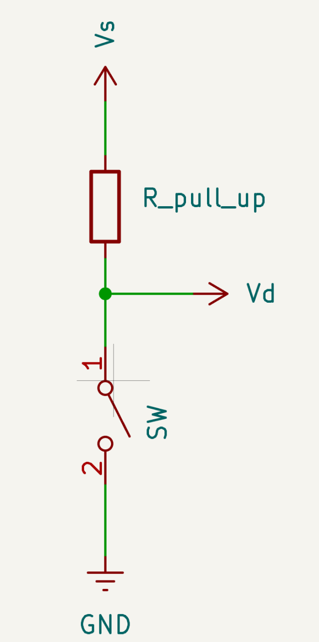
→ Interrupts (Covered in WS2)

Pull UP/DOWN resistors



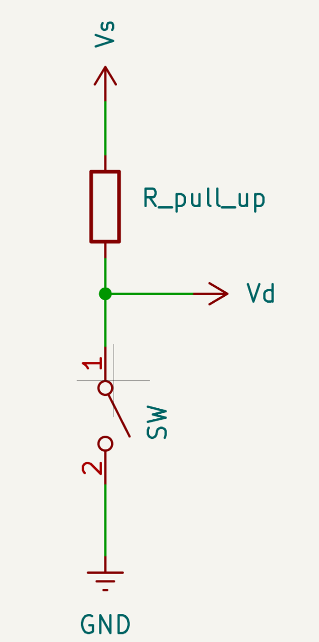


Pull-up resistors



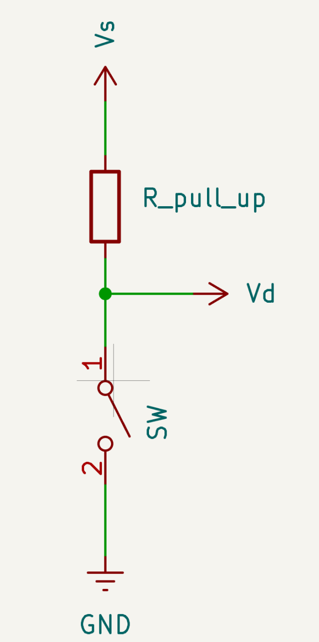
- Pull-up resistors ensure a wire is pulled to a logical HIGH in the absence of an input signal

Pull-up resistors



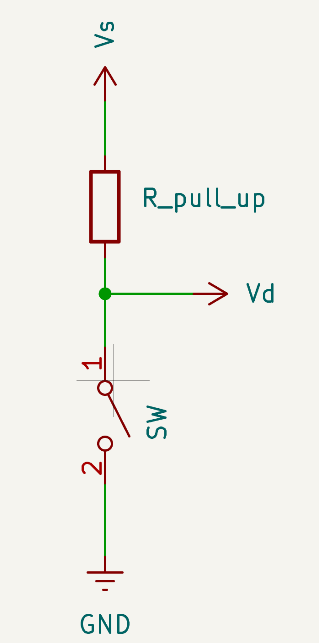
- Pull-up resistors ensure a wire is pulled to a logical HIGH in the absence of an input signal
- Digital logic circuits have three states: HIGH, LOW (when pin is OUTPUT), and FLOATING (in addition to HIGH/LOW when pin is INPUT)

Pull-up resistors



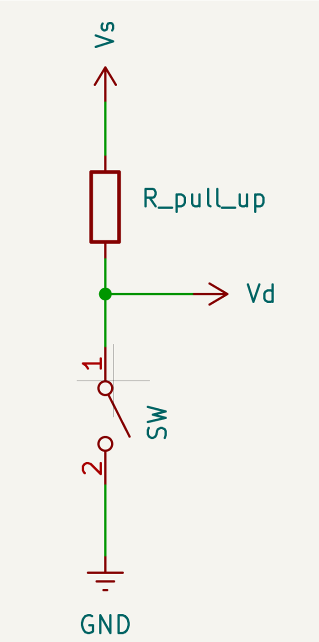
- Pull-up resistors ensure a wire is pulled to a logical HIGH in the absence of an input signal
- Digital logic circuits have three states: HIGH, LOW (when pin is OUTPUT), and FLOATING (in addition to HIGH/LOW when pin is INPUT)
- A floating input pin picks up noise → unreliable random state

Pull-up resistors



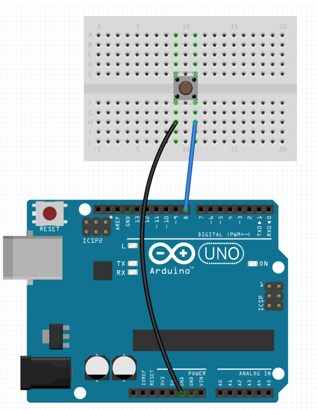
- Pull-up resistors ensure a wire is pulled to a logical HIGH in the absence of an input signal
- Digital logic circuits have three states: HIGH, LOW (when pin is OUTPUT), and FLOATING (in addition to HIGH/LOW when pin is INPUT)
- A floating input pin picks up noise → unreliable random state
- `INPUT` mode → high-impedance state (minimal current draw)

Pull-up resistors



- Pull-up resistors ensure a wire is pulled to a logical HIGH in the absence of an input signal
- Digital logic circuits have three states: HIGH, LOW (when pin is OUTPUT), and FLOATING (in addition to HIGH/LOW when pin is INPUT)
- A floating input pin picks up noise → unreliable random state
- `INPUT` mode → high-impedance state (minimal current draw)
- `INPUT_PULLUP` mode → pin tied internally to 5V (reads HIGH unpressed, LOW pressed)

Button example with `INPUT_PULLUP` I



☞ Connect the wiring diagram and upload the code `builtin_pullup.ino` Once the code is uploaded, open the Serial monitor and press the button. ▷ **Tap /**

Click on the pushbutton and monitor the output on the serial monitor

Using Interrupts

- Interrupts : enable the controler chip to stop the normal flow of a sketch and handle a task that requires immediate attention before continuing with what it was doing
- Can reduce the need for constant checking (polling). Reliable way to detect signals of very short duration.
- Handler for interrupts is called Interrupt service routine (ISR)

attachInterrupt(digitalPinToInterrupt(<pin >), <callback>, <trigger)
set an ISR to a pin with a callback function

<trigger> can be one of `HIGH,LOW,RISING,FALLING`

Millis and Micro I

Definition

returns the time elapsed since the code started in milliseconds ($10^{-3}s$)

returns the time elapsed since the code started in microseconds ($10^{-6}s$)

- Useful for fluidiying the code flow w/out `delay()`

Millis and Micro II

```
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000);
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000);
}
```

Millis and Micro III

Can you think of another way to write `blink` without `delay()`?

Hint : store value of millis() and micro() into a variable and compare it.

Millis and Micro IV

```
unsigned long time;

void setup(){
    pinMode(LED_BUILTIN,OUTPUT);
}

void loop(){
if(millis()-time > 1000){ //one second delay
    digitalWrite(LED_BUILTIN,!digitalRead(
   LED_BUILTIN));
    time = millis(); //update time
}
}
```

Less code and now more computing time to do other things !

Arduino libraries

the C programing workflow : header vs source code

Arduino libraries can be imported using the preprocessor directive `#include <library.h>` which tells the compiler to add the header code to the source code.

Library	**Purpose**	
<Servo.h>	Drive servomotors	
<Stepper.h>	Drive servomotors	
<sr04.h>	Interface sr04 distance module	
<EEPROM.h>	Use the Arduino's internal EEPROM	
<wire.h>	Interface devices with I2C	

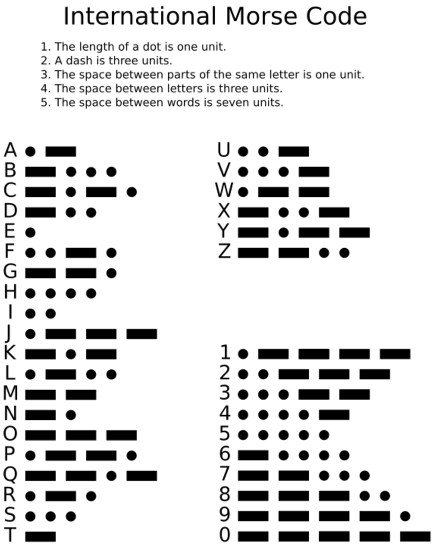
Table: Arduino libraries and usages

Arduino memory management

Memory	**Function**	**Use**
EEPROM	non-volatile memory	Configuration data
Flash	Read/Write ’hard’ memory	Executable code
RAM	Random Access Memory	Runtime and variables

Table: Arduino memories

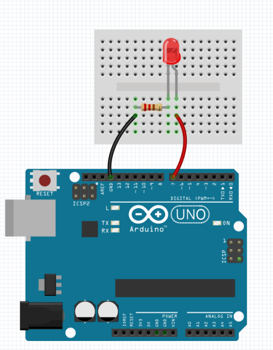
Morse code!



International Morse Code

1. The length of a dot is one unit.
2. A dash is three units.
3. The space between parts of the same letter is one unit.
4. The space between letters is three units.
5. The space between words is seven units.

A	● ▬	U	● ● ▬
B	▬ ● ● ●	V	● ● ● ▬
C	▬ ● ▬ ●	W	● ▬ ▬
D	▬ ● ●	X	▬ ● ● ▬
E	●	Y	▬ ● ▬ ▬
F	● ● ▬ ●	Z	▬ ▬ ● ●
G	▬ ▬ ●		
H	● ● ● ●		
I	● ●		
J	● ▬ ▬ ▬		
K	▬ ● ▬	1	● ▬ ▬ ▬ ▬
L	● ▬ ● ●	2	● ● ▬ ▬ ▬
M	▬ ▬	3	● ● ● ▬ ▬
N	▬ ●	4	● ● ● ● ▬
O	▬ ▬ ▬	5	● ● ● ● ●
P	● ▬ ▬ ●	6	▬ ● ● ● ●
Q	▬ ▬ ● ▬	7	▬ ▬ ● ● ●
R	● ▬ ●	8	▬ ▬ ▬ ● ●
S	● ● ●	9	▬ ▬ ▬ ▬ ●
T	▬	0	▬ ▬ ▬ ▬ ▬

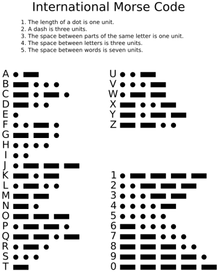


☞ Connect the following wiring diagram and complete the `morse_encoder.ino` code.

- **Open the serial monitor, and type in letters [aA-zZ] and numbers [0-9]**

Algorithm: Encoding Morse Code

```
Setup:
  Initialize Serial Communication

Loop:
  Listen for incoming Serial Communication
  If there is data
    Store the data in a string
    Call check()
    If check()
      Call morseToLED()

morseToLED(char[] signal)
  Takes an input signal (a char pointer array)
  Translates the signal into a combination
  of "Long" and "Short" signals

check()
  tries to match the input character with the morse ta
```

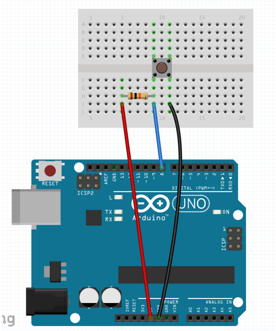


International Morse Code

1. The length of a dot is one unit.
2. A dash is three units.
3. The space between parts of the same letter is one unit.
4. The space between letters is three units.
5. The space between words is seven units.

A	● ▬	U	● ● ▬
B	▬ ● ● ●	V	● ● ● ▬
C	▬ ● ▬ ●	W	● ▬ ▬
D	▬ ● ●	X	▬ ● ● ▬
E	●	Y	▬ ● ▬ ▬
F	● ● ▬ ●	Z	▬ ▬ ● ●
G	▬ ▬ ●		
H	● ● ● ●		
I	● ●		
J	● ▬ ▬ ▬		
K	▬ ● ▬	1	● ▬ ▬ ▬ ▬
L	● ▬ ● ●	2	● ● ▬ ▬ ▬
M	▬ ▬	3	● ● ● ▬ ▬
N	▬ ●	4	● ● ● ● ▬
O	▬ ▬ ▬	5	● ● ● ● ●
P	● ▬ ▬ ●	6	▬ ● ● ● ●
Q	▬ ▬ ● ▬	7	▬ ▬ ● ● ●
R	● ▬ ●	8	▬ ▬ ▬ ● ●
S	● ● ●	9	▬ ▬ ▬ ▬ ●
T	▬	0	▬ ▬ ▬ ▬ ▬

☞ Let's program a Morse decoder withour arduino !



☞ Connect the following wiring diagram and complete the `morse_decoder.ino` code.

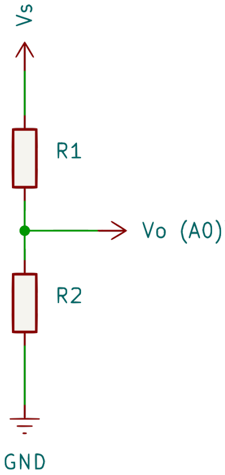
Algorithm: Decoding Morse Code

```
Setup:
  Initialize Serial Communication
Loop:
  If button is pressed (LOW)
    reset buttonPressLength
    updateUserSignal(press time)
    reset timeout
  If timeout > 2
    Encode current string
    check()
    Send decoded character (if any)

updateUserSignal:
  append corresponding signal 'S' or 'L'
  to user string

Check:
  Decodes a signal into a character
```

Exercise 2 : Voltmeter

Requirements

This exercise uses the concepts of Analog Readings and PHYS 142 electricity concepts.



Because the Arduino Analog Pins have a limited range of 0–5V on a 10-bit resolution scale, it is not possible to directly read voltages above 5V. To do so, we set up a voltage divider and work out a formula to find the Voltage Source (Vs) that we want to measure.

Exercise 2 : Voltmeter

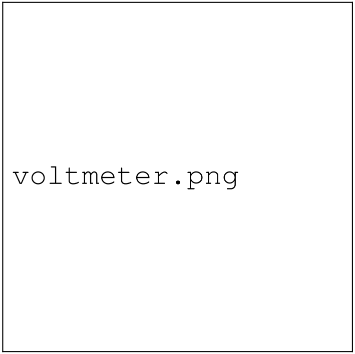


Figure: Voltmeter voltage divider

$$\text{loop 1:} \quad V_s - I(R_1 + R_2) = 0$$

$$\text{loop 2:} V_o - IR_2 = 0$$

$$V_0 = \left(\frac{V_s}{R_1 + R_2}\right) \times R_2$$

So solving for Vs

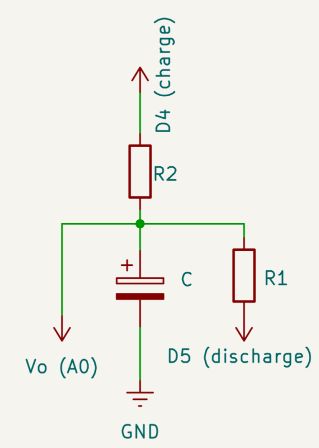
$$V_s = \frac{V_o\,(R_1 + R_2)}{R_2} = \boxed{V_o\left(\frac{R_1}{R_2} + 1\right)}$$

I is the unknown.

Exercise 2 : Voltmeter

```
#define Vo A0

int R1 = 100;
int R2 = 250;

void setup() {
  Serial.begin(9600);

}

void loop() {
  double Vs = analogRead(A0)*(R1/R2 +1);
  Serial.println('Voltage␣value␣is␣');
  Serial.print(Vs);
  Serial.print('␣V');

}
```

Exercise 3 : Capacitance meter



RC Time Constant: $\tau = R \cdot C$
where τ is the *time constant* of the circuit.

Arduino Pin Modes:

- `INPUT`: High impedance (almost no current flow), used for reading signals.
- `OUTPUT`: Low impedance, can be driven to $+5$V or pulled to ground.

Algorithm: Measuring Capacitance

```
Setup:
  Ask user for R
  Store R

DISCHARGE:
  Set Discharge Pin to OUTPUT LOW
  Set Charge Pin to INPUT
  Wait until voltage <  or timeout

CHARGE:
  Set Discharge to INPUT
  Set Charge to OUTPUT HIGH
  Wait until voltage ~ 0.632 × 5V or timeout
   = elapsed_time
  C =  / R
```

Charging

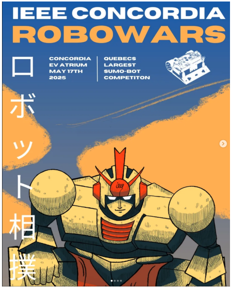
```
bool discharge(float epsilon = 5e-1, unsigned
   char timeout = 5){
pinMode(dis,OUTPUT);
pinMode(cha,INPUT);

int time = millis()/1000;
digitalWrite(dis,LOW);
while(1){
if(map(analogRead(vol),0,1023,0,5) < epsilon){
  return 1; // reached threshold
}
if((millis()/1000 - time) < timeout){
  return 0; // timeout
}}}
```

Discharging

```
double charge(float epsilon = 5e-1,unsigned
    char timeout = 5){
  pinMode(dis,INPUT);
  pinMode(cha,OUTPUT);

  int time = millis();
  while(1){
  if(abs(5.0*tau-(float)map(analogRead(vol)
   ,0,1023,0,5)) < epsilon){
  return (millis()-time)/R; //reached threshold
    of 63.2%(1-1/e) of E(5V)
}
if((millis()/1000 - time/1000) < timeout){
  return 0.0; // timeout
}}}
```

Any questions?

We need you!



We're interested in forming a team for the upcoming Robowars hackathon in May 2026!

contact me at ieee.competitions@mail.mcgill.ca

Questions? Comments? Suggestions? Threats? Insults?

Feedback Form

Workshop Interest Form

Further Reading I

Arduino Cookbook
O'Reilly, 3rd Edition, 2020.

Till Tantau.
The Beamer Class.
Available at: https://ctan.org/pkg/beamer

Arduino learning
Available at:
https://docs.arduino.cc/learn/microcontrollers/digital-pins/

Available at: https://eepower.com/resistor-guide/resistor-applications/pull-up-resistor-pull-down-resistor/#

Available at:
https://learn.sparkfun.com/tutorials/how-to-use-a-breadboard/all

Arduino Commands A–Z

2

A	`analogRead(pin),analogWrite(pin, value)`
D	`digitalRead(pin),digitalWrite(pin, value),delay(ms)`
M	`millis(),map(value, fromLow, fromHigh, toLow, toHigh)`
P	`pinMode(pin, mode),pulseIn(pin, value)`
S	`Serial.begin(baud),Serial.print(val),Serial.println(val)`