CPSC 1000: Introduction to Computer Science

Pulse width modulation (PWM) with Arduino

Robert Benkoczi, C556
robert.benkoczi@uleth.ca

23
~~16~~-Oct-2018
(Week 6)

Objectives

- Students will write Arduino programs that output a digital signal simulating an analog signal, using pulse width modulation on the Arduino PWM digital pins.
- Resources:
  - Arduino Notebook
  - Arduino Programming Language Reference

→ programming . . . easy (see Lab # 4): control lower power devices, eg: LED-s.

→ Part II : "analog signals" (PWM) for high power devices, such as motors.

Why analog signal output?

- Power source with adjustable voltage. DC motor speed = determined by the amount of power transferred from the electrical source (Ohm's law).

Analog signal : voltage is between low (0 V) and high (5V).

vs.

Digital signal : voltage is either low (0V) OR High (5V).

→ analog signal can transfer a variable amount of power to a component (circuit) such as a motor. (Motor: electrical energy → Kinetic energy).

$$\underset{\text{(physics)}}{\text{Power}} = \underset{\nearrow\ \text{voltage}}{U} \cdot \underset{\nwarrow\ \text{current}}{I} = \frac{U^2}{R} \quad \text{(proportional to voltage squared)} = \frac{\text{Energy}}{\text{time}}$$

Example : control the angular velocity of a DC motor using analog signal:

↓ direct current (as supplied by a battery)

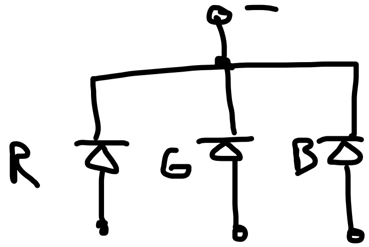
Voltage supplied on motor	Speed
x	v
$x' > x$	$v' > v$.

→ circuits generating analog signals tend to be more complex (cost of micro-controller ↗)

→ LED intensity : cannot be controlled with an analog signal ....

Why analog signal output?

- Set the colour of an RGB LED.



↳ we can generate "colour" by changing the relative intensity of the three LED's.

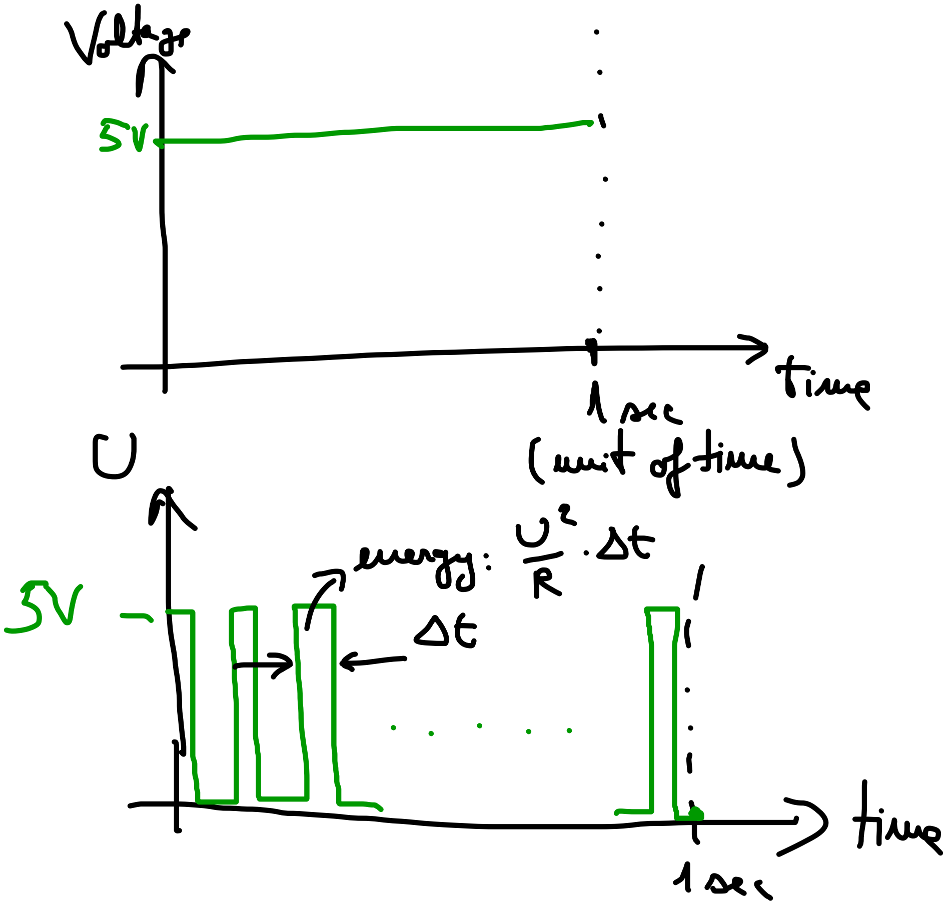
→ We have to use Pulse Width Modulation

Pulse width modulation

Arduino PWM frequency approx 490 Hz

→ idea: digital signal that transfers "arbitrary" amount of electric power to a component.

$$\text{Power} = \frac{\text{Energy}}{\text{time}}$$



a) Digital signal, high.

$$\text{Power} = \frac{U^2}{R} = \frac{\overset{5^2}{25}}{10\cdot10^3} = 2.5\ \text{mW}.$$

(10kΩ resistor)

b) Pulse with frequency f. (Frequency = # of pulses in unit of time)

$$\text{Power} = \frac{\text{Energy}}{\text{time}} = \underbrace{f\cdot\frac{U^2}{R}}_{\text{fixed}}\cdot\underbrace{\Delta t}_{\text{can be changed}}$$

Programming ✓

- No setup needed.
- `analogWrite(pin, value):`

See lab activity 4.

→ pin: digital pin, with label ~ in front of the pin number.

↳ pulse is generated by hardware components. The Arduino program can execute other tasks while the pulse is generated.

Question

Can we implement PWM using digital output?

↳ yes, we can write Arduino code to generate a PWM digital signal, but this will complicate the code (digitalWrite() & delay()).

Example

Choose the LED intensity randomly, and maintain it for 2 sec before choosing another intensity.

→ see lab Assignment 4.

Example 2

Set the PWM value proportional to the value of x relative to a range $m \leq x \leq M$.

↳ read potentiometer & generate "analog" signal proportional to the value read. (2nd question in Lab Assignment 4)

PART II

Controlling motors using the Adafruit motor shield v. 2.3

controlling high
power components.

Objectives

- Students will write Arduino programs that will control the speed of a simple 6V DC motor using an additional circuit: the Adafruit motor shield.
- Resources:
  - Arduino Notebook
  - Arduino Programming Language Reference
  - The Adafruit motor shield documentation: `https://learn.adafruit.com/adafruit-motor-shield-v2-for-arduino/library-reference`

Using the library:

To load the library definitions, types, and functions:

```
#include <Wire.h>
#include <Adafruit_MotorShield.h>
#include "utility/Adafruit_MS_PWMServoDriver.h"
```

- needed for the program to compile.
- insert at the top of the source code.

instruct the compiler to read additional files containing code for the shield.

To define the objects for the shield and motor(s):

```
Adafruit_MotorShield AFMS = Adafruit_MotorShield();
Adafruit_DCMotor *myMotor = AFMS.getMotor(4);
```

global variables.

variable that is a representation of the shield circuit.

global variable representing a DC motor connected to connector labelled M4.

motor connector, eg: M4. Use 1 for M1 etc.

How do we set the rotation speed, direction of rotation, or state of the motor (ON/ OFF).

→ call functions → run on the motor variables.
↘ setSpeed

Example

a) myMotor -> run (state);

dash , greater than

state = → RELEASE (turn motor off)
→ FORWARD (turn on)
→ BACKWARD
(turn in opposite directions).

b) myMotor -> setSpeed (value);

$0 \leq \text{value} \leq 255.$