ROBa: labs

Jan Beran
Daniel Bambušek

Faculty of Information Technology
Brno University of Technology
Božetěchova 1/2. 612 66 Brno – Královo Pole

November 8, 2022

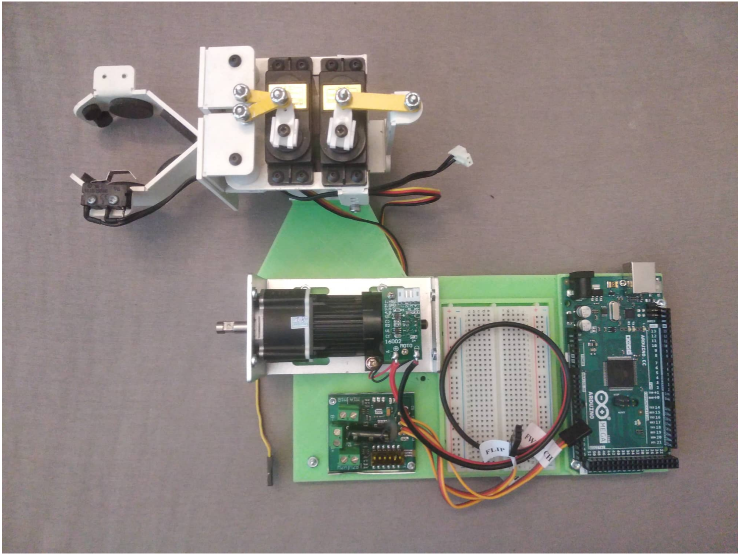
Overview

- **Lab1: Arduino basics**
- **Lab2: Sensors**
- **Lab3: Motors and actuators**
- **Lab4: ROS basics**
- **Lab5: Getting advanced in ROS**
- **Lab6: Surprise**

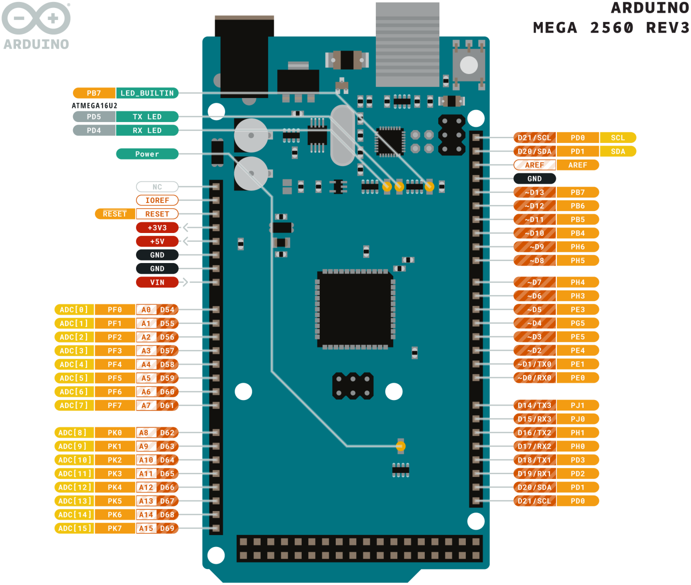
Arduino basics: overview

- How to work with Arduino, lab kit, breadboard, etc.
- Arduino IDE
- Blinking LED, Serial communication, PWM, `analogRead` and `digitalRead`...
- Next time: I^2C , sensors

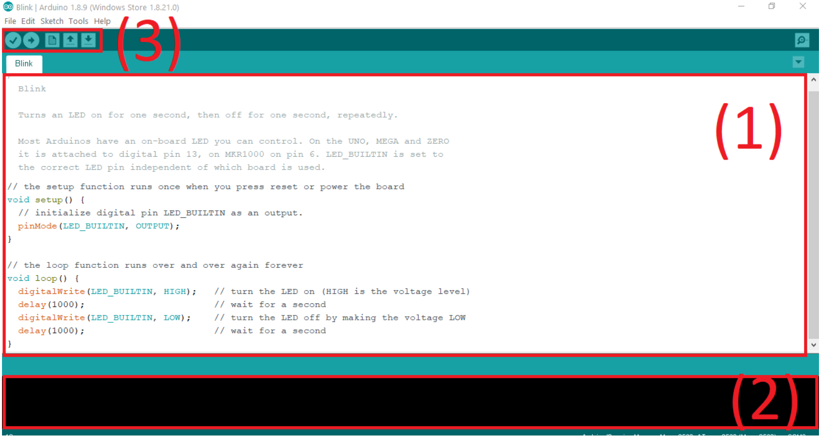
Lab kit, breadboard



Arduino MEGA 2560



Arduino IDE



Blink | Arduino 1.8.9 (Windows Store 1.8.21.0)
File Edit Sketch Tools Help
(3)
Blink
(1)
(2)

```
/*
  Blink

  Turns an LED on for one second, then off for one second, repeatedly.

  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
  the correct LED pin independent of which board is used.
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

1. Place for your code
2. Debug info
3. Build, flash, new file, open file, save project

Example 1: Blinking LED

1. Desktop –> Lab1 –> Blink –> Blink.ino
2. Upload with → arrow

Task1: PWM

- Task: Continuously change the brightness of LED from `0` (full off) to 255 (full on)
- No analog output, PWM instead
- You can find a template in `PWM` folder
- Signature: `analogWrite(pin, value)`

Task2: Serial communication, inputs

- Task: Read both digital and analog data from a given pin connected to potentiometer and send the data over Serial line to your computer
- Use template `AD-read.ino` from the folder of the same name

Task2: Serial communication, inputs

- Serial communication is established with `Serial.begin(baudrate)`
- Sending is done via `Serial.print(something)` and `Serial.println(something)`
- `digitalRead(pin)` returns just 0 or 1 (1bit ADC :))
- `analogRead(pin)` uses 10bit ADC –> values from 0 to 1023

Task2: Serial communication, inputs

- On pin `A0` there is a potentiometer
- Use both `analogRead(pin)` and `digitalRead(pin)` to obtain analog and digital version of the same input
- Print them via `Serial.print()` or `Serial.println()`
- Sending is done via `Serial.print(something)` and `Serial.println(something)`

Task3: Sonar HC–SR04

- HC–SR04: simple version of SRF–08 from next lab2
- Sonar: working principle
- HC–SR04 working principle:
  - Set `trigger_pin` to HIGH for 5 microseconds then set LOW
  - HC–SR04 sends ultrasonic burst and sets `echo_pin` to HIGH
  - When the burst returns, HC–SR04 sets `echo_pin` to LOW
  - => length of pulse on `echo_pin` == how long sound needs to reach the obstacle and return back

Task3: Sonar HC–SR04

- Task: Trigger HC–SR04 and measure the length of pulse on `echo_pin`
  - For the first part, complete `set_measurement` function
  - For the second part, use `pulseIn(pin, HIGH)`
  - Speed of sound is 0.0343 cm/us

End of lab1

Sensors: overview

- Review of lab1 (Arduino basics)
- I^{2}C
- SRF−08
- IMU
- Next time: motors

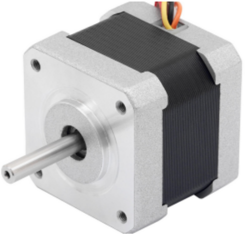
End of lab2

Motors: overview

- Review of lab1 (basics)
- Motor types and how to control them
- Servos and pulse control
- DC motor: transistor
- DC motor: H–bridge
- DC motor: H–bridge again
- DC motor: Sabertooth 2x5 RC
- Next time: ROS

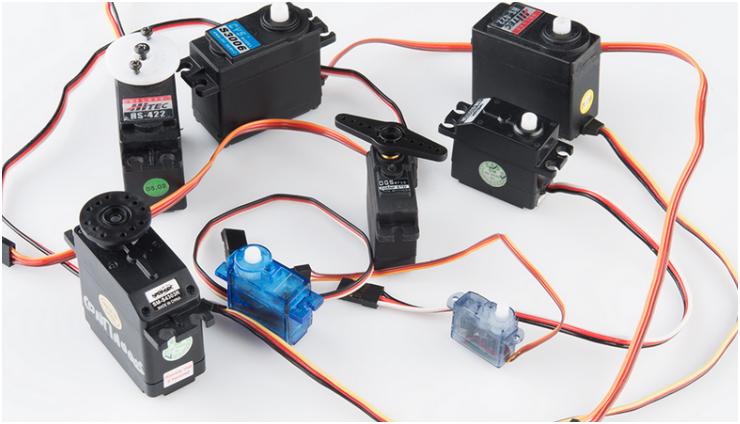
Motor types

Type	Control mechanism	Real way of controlling	Usage and notes
Servo	1–2 ms pulse every 20 ms	Controller boards Directly using libraries Directly–manually	Positioning (can hold angle) Usually cannot move more than 180 (360) degrees
DC (many types)	By voltage Difficult (BLDC)	Directly (brushed) H–bridges (brushed) Controller boards (both)	Wide; wheel power, propeller power...
Stepper	Difficult	Controller boards	Precise movement (3D printers)

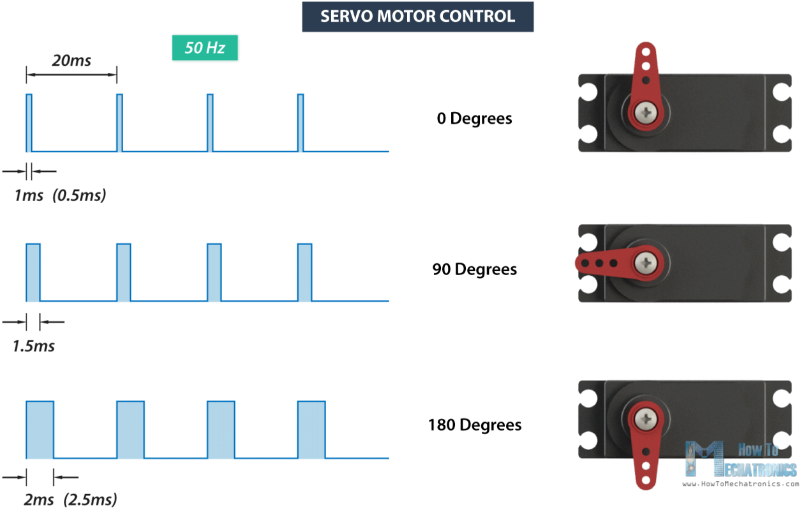


Servos

- Used in robotic arms etc.
- Possible to set exact angle
- Simple controlling (some electronics inside servo)



How to control servo motors



Task 1: Controlling servo

- Open `servo.ino`
- Complete `servo_write()` function
    - You might need `delayMicroseconds()` function
- Try to grip something :)

DC motor: control with transistor

- Simple: transistor as a switch
- Cannot control direction :(

Task 2: Controlling DC motor with a transistor

- Open `motor1.ino`
- Use `analogWrite(motor_pin, value)` to control the motor
- Gradually increase the speed of the motor and then decrease it again

How to control direction?

DC motor: H–Bridge

- More complex, but still simple
- Can control direction :)
- Can damage your motor, PSU or MCU if misused

Task 3: Controlling DC motor with H–Bridge

- Open `motor_hbridge.ino`
- Get familiar with the code (serial line protocol...)
- Gradually increase the speed of the motor and then decrease it again

Task 3: Recap

- It worked! :)
- But:
  - Many parts
  - Need of electronic knowledge
  - Takes space
  - Takes time to make
  - Unreliable
- So...

L293D

- Two H–bridges (four half H–bridges :))
- `https://www.ti.com/lit/ds/symlink/l293.pdf`
- Details in datasheet

Task 4: Controlling DC motor with L293D

- Open `motor-l293D.ino`
- `motor_pins` control direction according to the datasheet
  - `control_pins[0]` corresponds to 1A
  - `control_pins[1]` corresponds to 2A
- `enable_pin` controls speed with `analogWrite()`:
  - `0`: no speed (stop)
  - 255: max speed
- Task: try to change speed and direction:
  - Set one direction and gradually change the speed from stop to maximal speed and back to stop
  - Change the direction and do the same

Sabertooth 2x5 RC

- Control board for motors
- Same logic as with servos:
  - 1–2 ms pulses every 20 ms
  - Setting speed and direction instead of angle

Task 5: Control DC motor as a servo (almost)

- Open `sabertooth.ino`
- Complete `sabertooth_write` function
- Motor should perform fluent speed and direction change

End of lab3

Sensors: overview

- Review of lab2 (sensors)
- Types of motors and how to control them
- Encoders
- Servos and pulse control
- DC motor: transistor
- DC motor: H–bridge
- DC motor: H–bridge again
- DC motor: Sabertooth 2x5 RC
- Next time: ROS

-

-