Introduction to Robotics

Lecture 2: Wiring Digital Circuits & Building the Robot

24. 9. 2018

ParaDiSe

Today Goal

- get (roughly) familiar with the components
- build the mechanics of the robot
- wire it all together
- test it all using a simple program (via Arduino framework)

Next week goal:

- learn about bare-metal CPU in depth
- learn the difference between Arduino & classical programming
- dump the Arduino framework and start writing a proper firmware

Preliminaries

What Is a Microcontroller

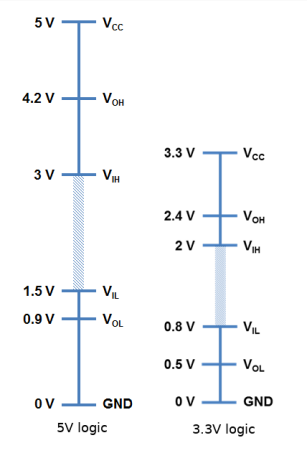
Microcontroller (MCU) is a small computer on a chip usually consisting of:

- a computational core (with ALU, memories, etc.),
- a set of HW peripherals,
- from which one of them is general purpose I/O (GPIO)

$\rightarrow$ MCU is capable of direct, low level interaction with the environment.

$\rightarrow$ the skill of MCU programming is in mastering the peripherals

Logic One and Zero



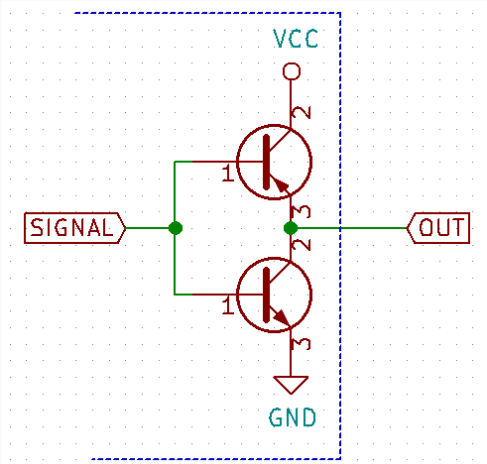
(based on SparkFun images)
https://learn.sparkfun.com/tutorials/logic-levels/arduino-logic-levels

Digital signals are usually encoded as voltage:

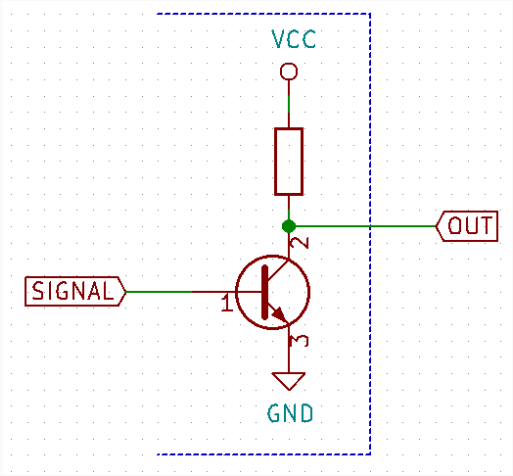
- all the voltages are referenced to a common potential (usually denoted as ground, GND)
- commonly used 5V, 3.3V systems
- more and more popular: 2V, 1.8V, 1.5V and 1V systems
- dead region – prevent signal ringing on edges

Model of Digital Output – Advanced

Push pull

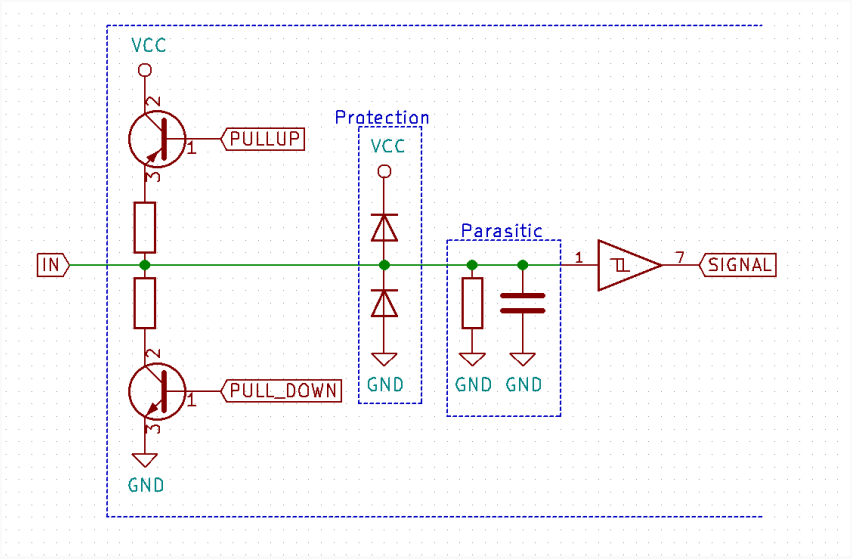


Open collector



An output can usually sink about 20mA.

Model of Digital Input – Advanced



Wiring Digital Circuits

Dos:

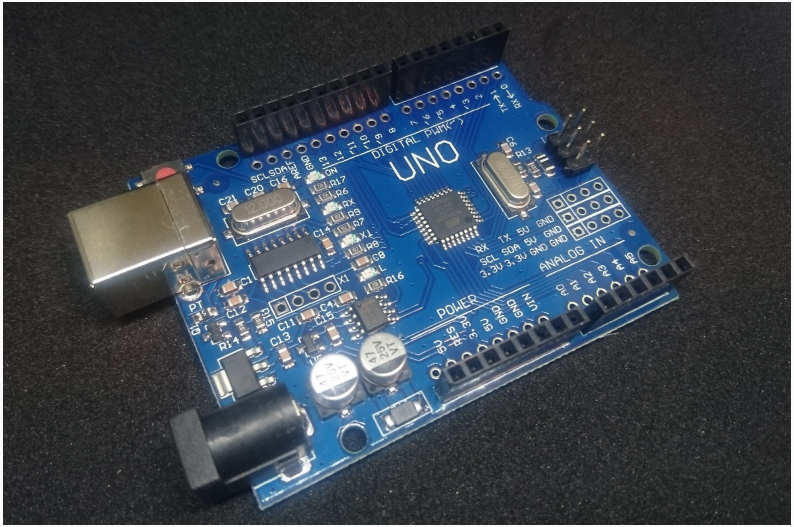
- connect inputs together
- connect input to output
- connect open collector outputs
- connect grounds together
- connect the same VCCs together

Don'ts

- connect push-pull outputs together
- connect outputs to power rails
- exceed the rated voltage

Components

Arduino Uno



Arduino Uno

- development kit for Atmega328P MCU aiming for beginners
- built-in firmware flasher
- different (simpler) pin labeling compared to a bare MCU

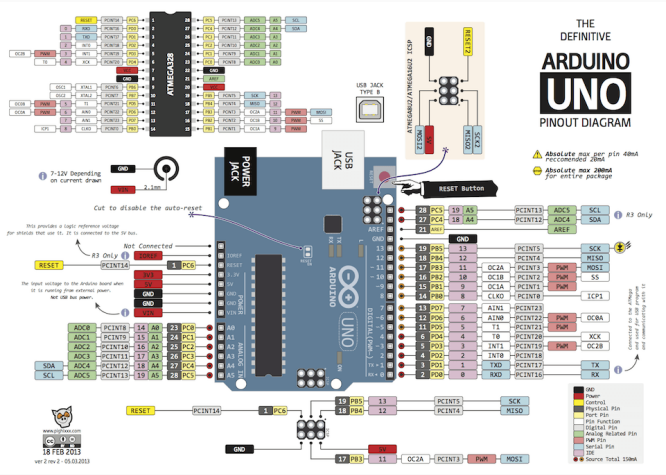
Atmega328P

Datasheet:

`https://www.mouser.com/pdfdocs/Gravitech_ATMEGA328_datasheet.pdf`

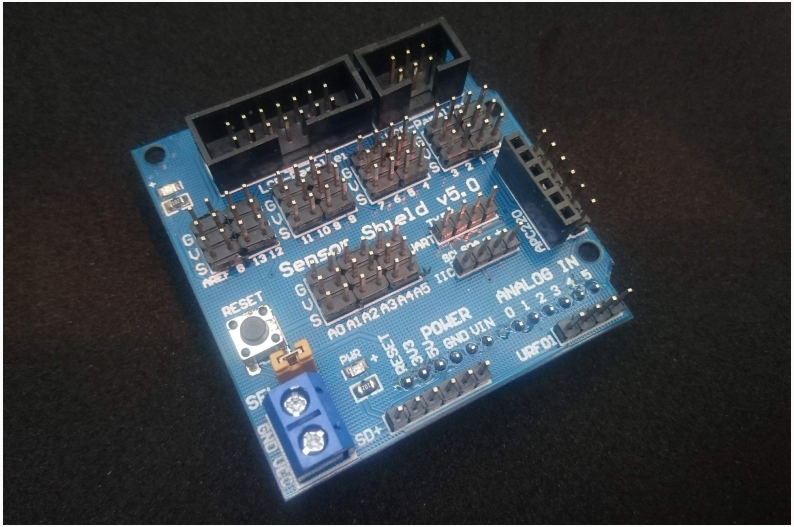
- 8-bit MCU
- 16 Mhz, 2 KBytes of RAM, 32 KBytes of flash
- 23 GPIO
- timers, UART, SPI, I2C, ADC
- can be programmed in ASM, C, C++, (Haskell, Rust)

Arduino Uno Pinout



http://marcusjenkins.com/arduino-pinout-diagrams/

Sensor Shield

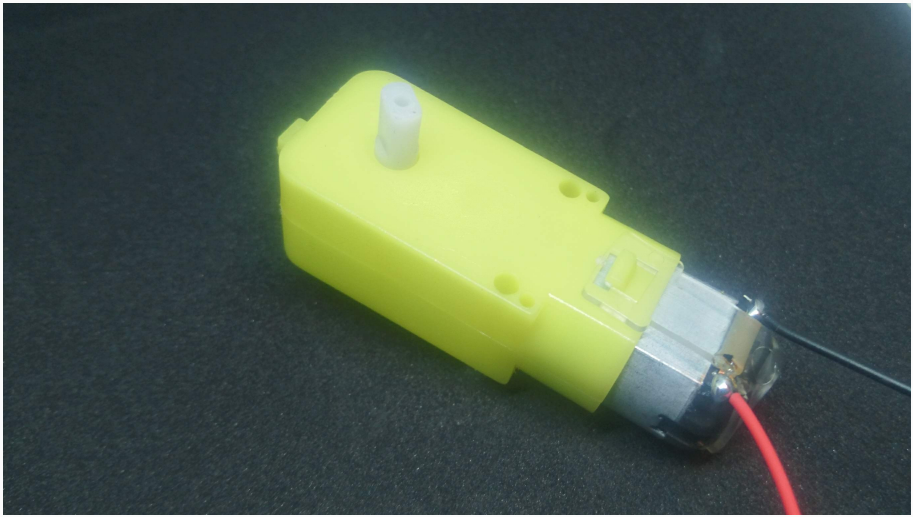


Sensor Shield

"There's not enough GND and VCC pins. Never, ever." – folk wisdom

- expands the pins of Arduino so that for every GPIO there is:
  - GPIO pin
  - VCC
  - GND
- break outs standard connectors for:
  - LCD
  - Bluetooth module
  - RF module
  - SD card
  - etc.

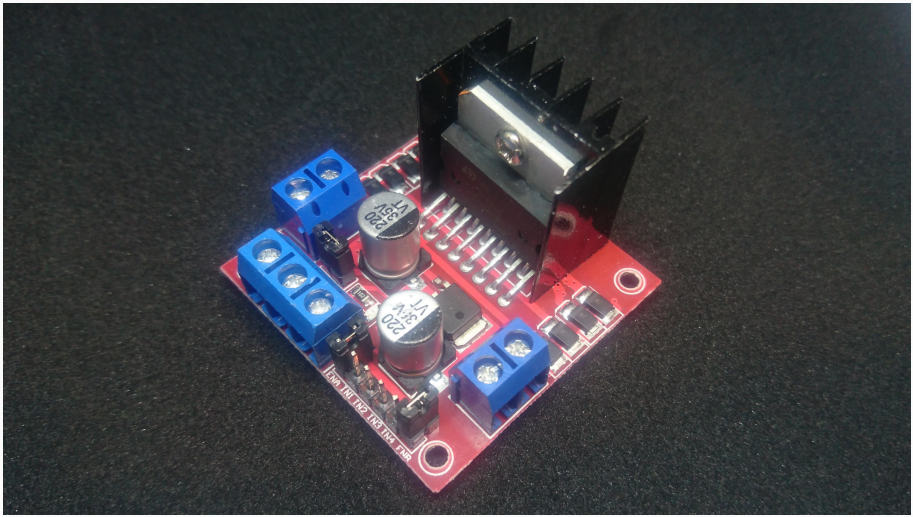
Motor with Gearbox



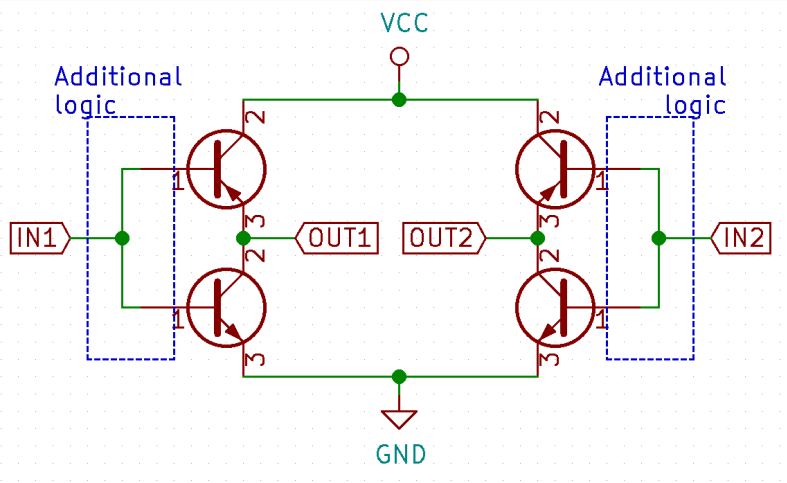
Motor with Gearbox

- rated voltage 7V
- stall current 1A
- 120 RPM
- polarity of the leads determine direction of rotation
- smaller voltage, lower RPM (but not linear)
- no feedback – with higher load, lower RPM

Motor Driver



Motor Driver – What is it?



Motor Driver

Datasheet:

`https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf`

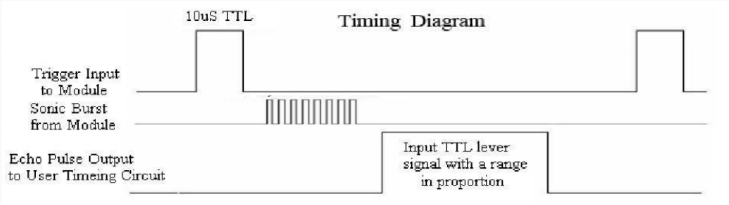
- allows us to drive motor by MCU pins
- 2A, up to 40V

Ultrasonic Sensor HC–SR04

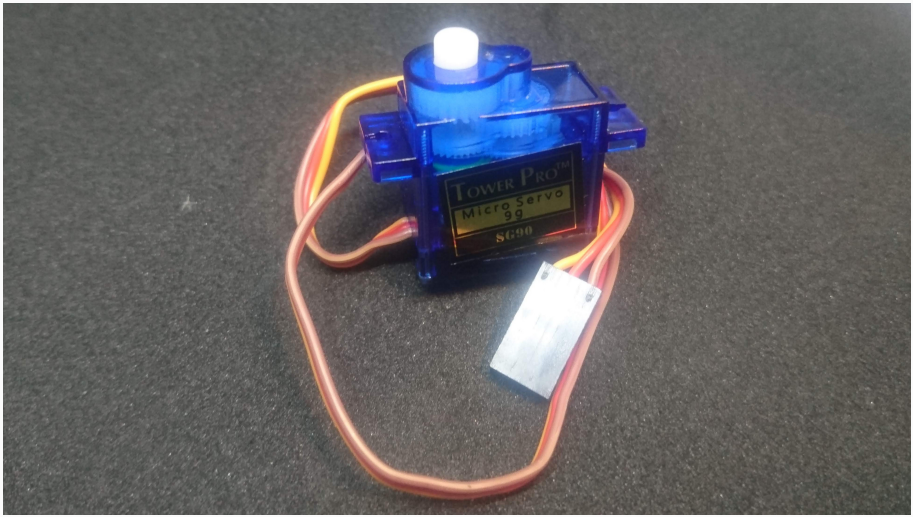
Ultrasonic Sensor HC–SR04

Datasheet: `https://www.mouser.com/ds/2/813/HCSR04-1022824.pdf`

- measures distance by sending a sound wave and listening back
- powered by 5V and 5V compatible
- measurement:
  - put `trig` high for at least 10 us
  - measure pulse with on `echo`
  - distance = pulse width · speed of sound / 2



RC Servomotor SG90



RC Servomotor SG90

- rotates ± 70 degrees
- powered by 5V
- controlled by PWM at 50 Hz:
  - 1.5 ms = neutral position
  - 1 ms = max left
  - 2 ms = max right
- no feedback to the controller (best effort)
- wiring
  - brown = GND
  - red = 5V
  - oragne = pulse

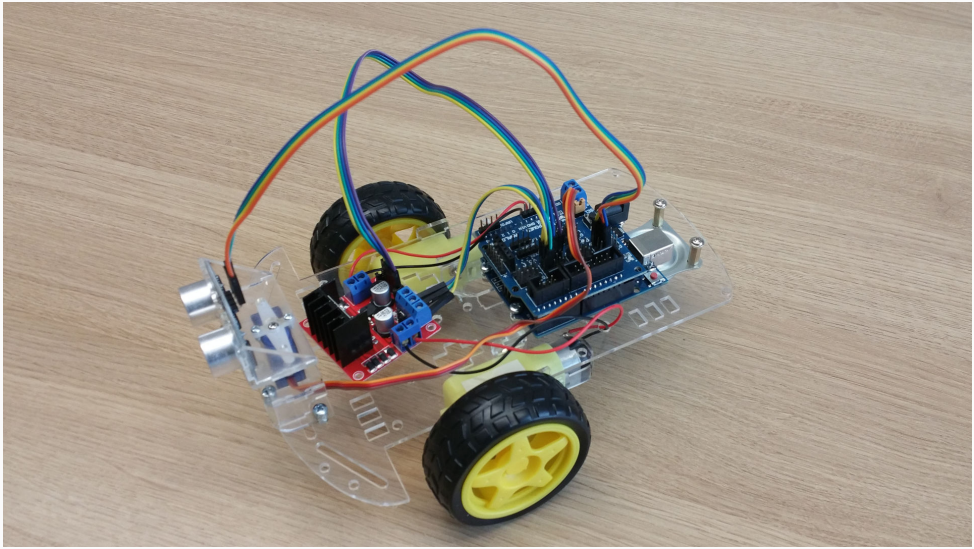
Building the Robot

Instructions

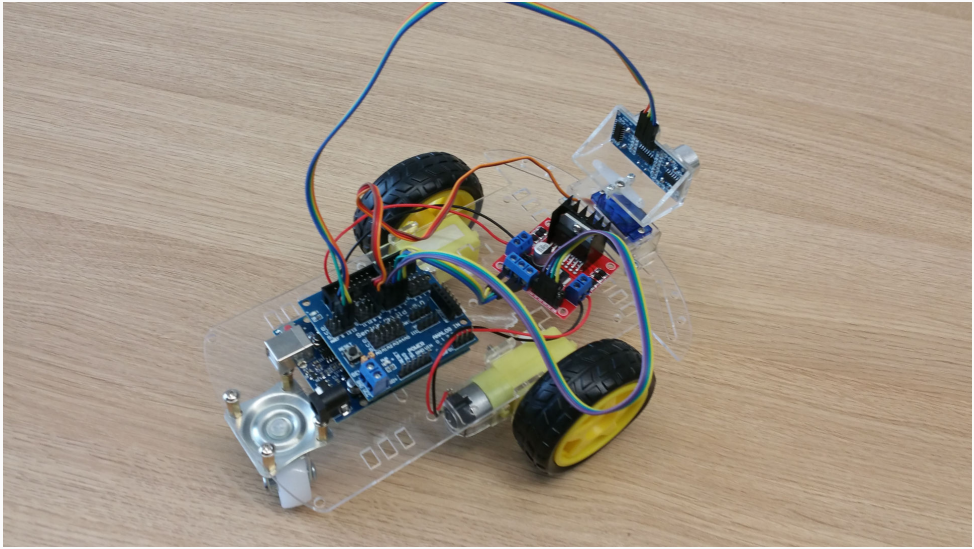
- the order of the assembly does not matter (use common sense)
- all part are in the bags
- follow the images
- wire it (all GPIO are for our purposes equal except GPIO0 and 1 – avoid them)

Mechanical tunings are welcomed!

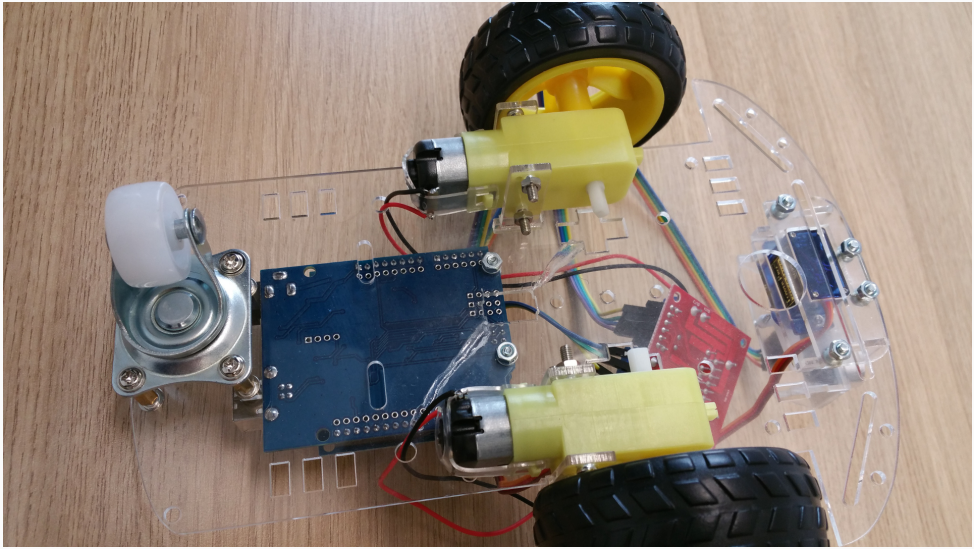
Overall View – Front



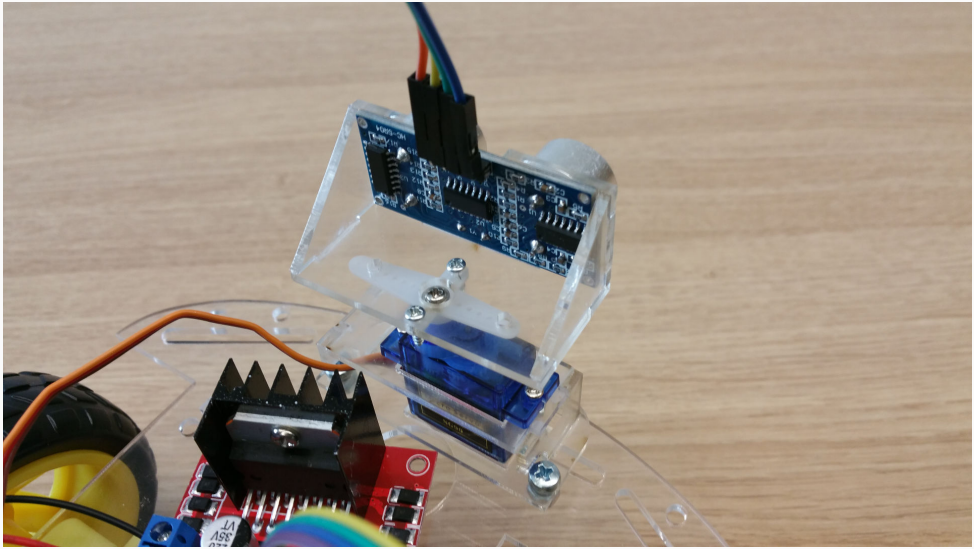
Overall View – Back



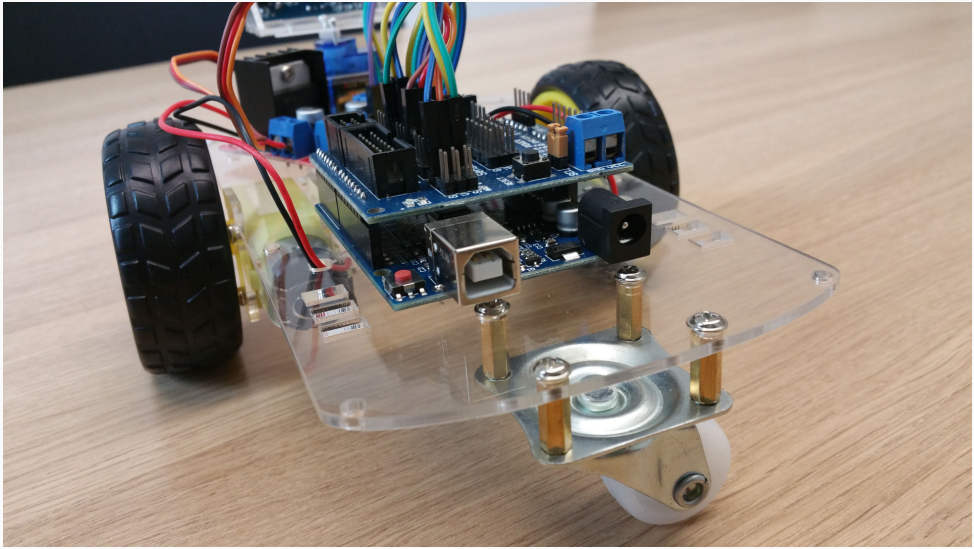
Bottom View



Eyes Detail



Rear Wheel



Programming

The Arduino Framework

"Everyone should be able to code and build HW" – `https://www.arduino.cc/`

- several HW modules with simple interface (but limited capabilities)
- C++ libraries for easy use (dumps many C++ features)
- good for:
  - hobbyists
  - quick prototyping
  - many libraries for external HW
- problems:
  - not so intuitive for experienced programmer (many antipatterns)
  - cannot use all the features of MCU (especially asynchronosity)
  - quite high overhead

Our Viewpoint on Arduino

- good for hardware validation & motivation
- hides something we want to learn

- use it for this lecture
- we will show alternative in the next lecture
- since then, it is up to you if you continue to use Arduino

PlatformIO

- CLI tools for easy management of embedded toolchains, frameworks & libraries
- GUI plugins for many editors (VS Code, Atom, Emacs...)
- provides automatic build system with multi-platform support

Why to use it:

- simplifies toolchain setup
- manages repository of libraries and allows for easy dependency specification

Why not to use it:

- you enjoy writing cross-platform makefiles and it takes the joy away from your

Hello World

```
#include <Arduino.h>

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

void loop() {
    Serial.println( "Hello world!" );
    delay( 500 );
}
```

- start a new project in PlatformIO
- choose Arduino Uno as the board
- choose Arduino as the framework

- build it by ctrl + b
- upload it by ctrl + u
- view serial terminal by ctrl + s

Arduino reference: `https://www.arduino.cc/reference/en/#functions`

Working With GPIOs

```
#include <Arduino.h>
const int LED_PIN = 13;
void setup() {
    Serial.begin( 9600 );
    pinMode( LED_PIN, OUTPUT );
}

void loop() {
    digitalWrite( LED_PIN, LOW );
    delay( 300 );
    digitalWrite( LED_PIN, HIGH );
    delay( 300 );
}
```

```
#include <Arduino.h>
const int BTN_PIN = 13;
void setup() {
    Serial.begin( 9600 );
    pinMode( BTN_PIN, INPUT_PULLUP );
}

void loop() {
    if ( !digitalRead( BTN_PIN ) ) {
        Serial.println( "Pressed" );
        delay( 500 );
    }
}
```

Working With Servos

```
#include <Arduino.h>
#include <Servo.h>
Servo servo;
void setup() {
    Serial.begin( 9600 );
    servo.attach( 9 );
}
void loop() {
    servo.write( 0 );
    delay( 500 );
    servo.write( 180 );
    delay( 500 );
}
```

- moves the servo from min to max
- instead of begin, you have to attach (stupid)
- write accepts angle in degrees (0–180)

Task 1 – Turn the Motors

Baby steps; write a simple program that:

- turns the wheels to the both directions
- controls the speed of the wheels (`analogWrite` might help you)

Final Task: Write a program that allows you to control the robot using the WASD keys on your computer.

Task 2 – The Ultrasonic Sensor

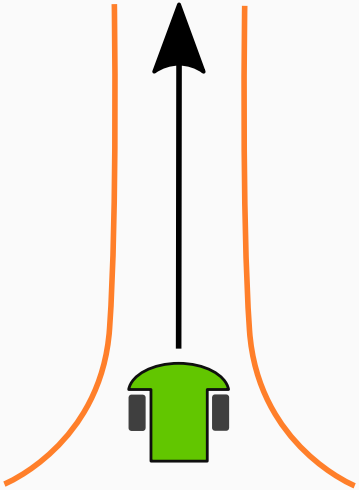
Write a simple program that repeatedly prints to the serial terminal distance in centimeters from the sensor.

- read the datasheet (link previously in the slides) to see the measurement procedure
- `pulseIn` might help you
- speed of sound in the air at 25 C is $34 \frac{\text{cm}}{\text{ms}}$

Then test the sensor

- does it work with small objects?
- how does it behave with soft/fluffy stuff (e.g. a soft toy)
- how does it behave in corners?
- how does it behave when two sensors look at each other?

Task 3



Program the robot such that:

- given a long corridor the robot can go through the corridor not hitting the walls
- it is not hard-coded to a fixed corridor width
- try to achieve smooth motion
- hint: you have an ultrasonic sensor on rotational mount

We will compare our approaches in the next lecture.