FABLAB
SAUÐÁRKRÓKUR

A brief introduction to making electronics

by Hans Peter Sommer Haastrup-Nielsen

Class that was held in November in Fab Lab Sauðárkrókur

A brief introduction to making electronics

So you want to create an electronics project?

That's great! However, if you are new be advised – it WILL take some time. Instant gratification is a bitch – but on the other hand you will be more proud when you complete a long project!

Alright, let's go! First – what do you want to make? **Try to describe it with words.** For example 'I want to make a self-watering flower pot'.

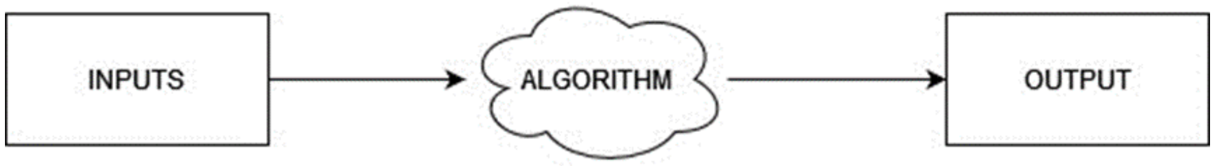
Now, try to define the elements needed for such a system – what would be needed to create a project like you described:

I need:

- A place to hold my plant (should be watertight).
- A way to measure if the plant needs water.
- A place to store water.
- A way to get the water from the storage to the plant.
- An indicator if I have no more water.
- A sensor for the amount of water left.

The list you just created is the main part of your system.

All systems can be described using the following diagram:



Going from right to left you will have some outputs controlled by an algorithm that decides on what to do based on the sensor inputs. Let's try to move our list from above into this methodology:

- Inputs
    - A way to measure if the plant needs water.
    - A sensor for the amount of water left.

- Outputs
  - A way to get the water from the storage to the plant.
  - An indicator if I have no more water.

Some parts of the system are **mechanical parts** that we will need to manufacture somehow. The planter could be a ready-made pot and the container for water could be a bottle. We will not discuss these parts further in this project.

Finding sensors

Okay, so we need to find a way to measure if a plant needs water – let's google that: 'measuring if a plant needs water'. First result states something like 'the easiest way to check if your plant needs water is by sticking your finger into the soil'. We want to know how wet the soil is – we need a **Soil moisture sensor**!

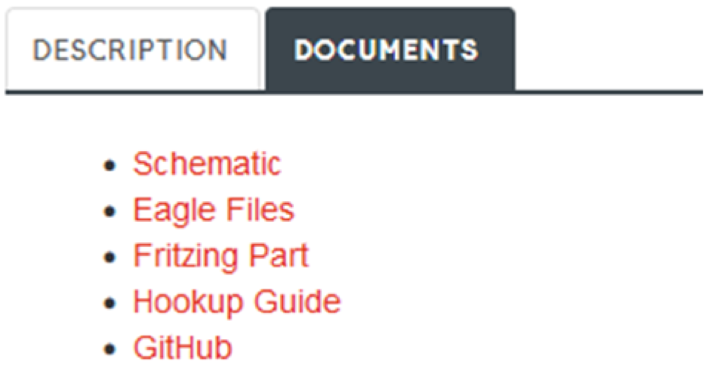
For the water left in the tank – we need a water level sensor.

For getting water to our plant – a pump.

The last thing, an indicator – we want a light, an LED.

Now, where to find sensors? Let's go to https://www.sparkfun.com/categories/23

In the menu we can select a type of sensor to filter out the one we are looking for – there is also a search function. Try searching for 'Soil'. There are multiple hits – which one to choose? Here it get's a little complicated but you could search for the cheapest? When you are not sure you might not get the right sensor the first time but then you know what NOT to look for the next time. That's called a learning experience 😉



All sensors on sparkfun will have a guide on how to connect them to a controller:

Click on 'Documents' and 'Hookup guide'. This will guide you on how to get a reading from your sensor!

This will usually be a quite extensive guide with both diagrams, explanations and code for your controller to measure the sensor along with help on calibration and much more information.

Using the same approach we can find a water level sensor. This sensor will most likely act like a button so it will not give you the amount of water left but merely and indication of IF there is water left.

Similarly the pump is a digital OUTPUT from your controller – you can turn it on or off. You might need a relay or a transistor to control the current but it will still be a single digital signal for control

The last part – the indicator for water in the tank is also a digital output. We can search for LED (and get a LOT of results) – Choose one or the other. It will be a matter of personal preference for which color you like the most to represent the warning light.

Good luck with your project!

Remember: How do you eat an elephant? In a lot of small bites!

Inputs & outputs and how to control almost anything! - by Hans-Peter

Introductory class for making your own electronic designs.

Administrative information

· 3 classes, 3 hour duration

· Homework to be completed between classes

· Hardware platform developed and made available should belong to the students during and afterwards

The classes will try to cover as much material as possible in a very short timeframe. There will be a few homework assignments and a few extra tasks for the adventurous.

Contents of the classes

Class 1 – Introduction, sensor technology in general, signals, programming and protocols.

Class 2 – Wifi, Signal conditioning, Windows forms programming (C#), how to use electronic modules as building blocks

Class 3 – how to read a datasheet, controlling outputs, Schematics and layout, PCB production

If you are doing this for the first time, skip the blue text, you will get there later.

Assembly and initial programming of platform boards

This document describes the steps needed to prepare for the electronics class. It also contains links to all necessary extra teaching materials and sources needed during the class.

Preparations

A platform has been developed for the classes. This platform is supposed to be the property of the student during and after the class sessions. Therefore we need to be able to make them per request.

Step one – assemble the platform. This is fairly straightforward but not without its own caveats.

· Start with the USB bridge – CP2102. This is a small QFN package and you might need to use the 'drag-soldering' technique in order to put it on the board.

  o Start by adding solder to a single pin.

  o Orient the package so that pin 1 matches the silkscreen indication

  o Heat up the solder on the single pin while holding the package in place with a pair of tweezers.

  o Check alignment on all four sides

  o When happy with alignment, add solder to the OPPOSITE side from the initial single pin. This will decrease the risk of accidental desoldering the part.

  o Solder the perpendicular sides of the package

  o Do the side with the single pin last.

· Add components around the USB bridge – in order to test the bridge all parts need to be soldered.

  o The RX/TX LEDs need to be activated from a program called 'Simplicity Studio' available from Silicon Labs website. This sounds complicated but is quite straightforward to do. I recommend that this is done as a last step since all boards can be done at the same time and the program can be a bit unstable.

o In Simplicity studio the max allowed power draw on the bridge can also be set.

§ Set this to 500mA and configure the alternate functions of GPIO0 and GPIO1 to RX and TX Toggle respectively.

· Add the rest of the SMD components

· Add buttons, MCU and resonator

· Add pins for ICSP and UART jumper

· Add Headers for ESP32 and eventually all other pins and headers you want to add

· Add the potentiometers last, they have an annoying height.

Programming the bootloader

The MCU needs to have a bootloader programmed before it can be recognized in the Arduino IDE.

Depending on which MCU you get this can be more or less complicated. Welcome to the deep end – Here be dragons!

· First, find a working Arduino – or an ISP if you have one.

· Open the example called 'ArduinoISP' (example 11 in the built-in list)

o Program this to the Arduino that you have and leave it connected to the computer. We will use the same serial port later

· Identify which MCU chip you have. If you have an 'ATMEGA328' you need to add a custom entry to the Arduino Boards file:

o Open the boards.txt (in folder C:\Program Files (x86)\Arduino\hardware\arduino\avr)

§ You need to have administrator rights to be able to save this file!

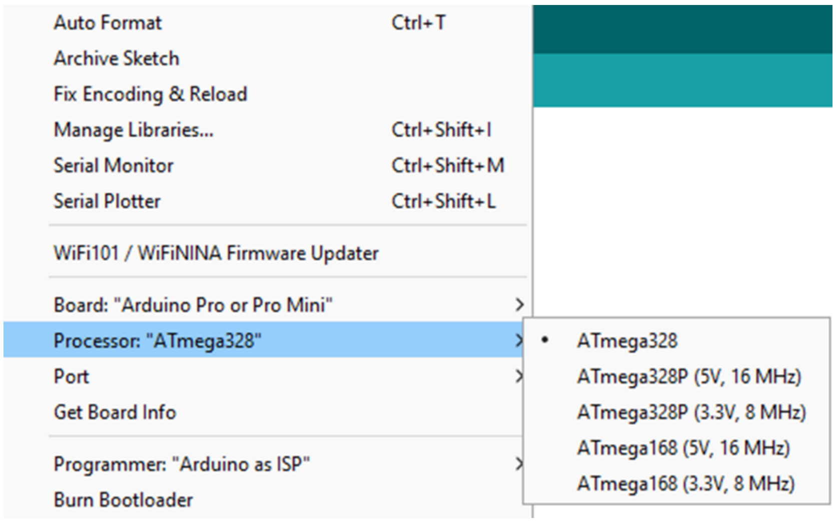
o Search for the 'Arduino Pro or Pro mini section'

o Insert the following

```
## Arduino Pro or Pro Mini w/ ATmega328

## ------------------------------------------------

pro.menu.cpu.16MHzatmega328n=ATmega328


pro.menu.cpu.16MHzatmega328n.upload.maximum_size=30720

pro.menu.cpu.16MHzatmega328n.upload.maximum_data_size=2048

pro.menu.cpu.16MHzatmega328n.upload.speed=57600


pro.menu.cpu.16MHzatmega328n.bootloader.low_fuses=0xFF

pro.menu.cpu.16MHzatmega328n.bootloader.high_fuses=0xDA

pro.menu.cpu.16MHzatmega328n.bootloader.extended_fuses=0xFD

pro.menu.cpu.16MHzatmega328n.bootloader.file=atmega/ATmegaBOOT_168_atmega328.hex


pro.menu.cpu.16MHzatmega328n.build.mcu=atmega328

pro.menu.cpu.16MHzatmega328n.build.f_cpu=16000000L
```

o The only change here is the MCU type but since this requires all entries to be unique we need to add all the lines.

o Save the file and open the Arduino IDE



Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino Pro or Pro Mini"
Processor: "ATmega328"
Port
Get Board Info
Programmer: "Arduino as ISP"
Burn Bootloader

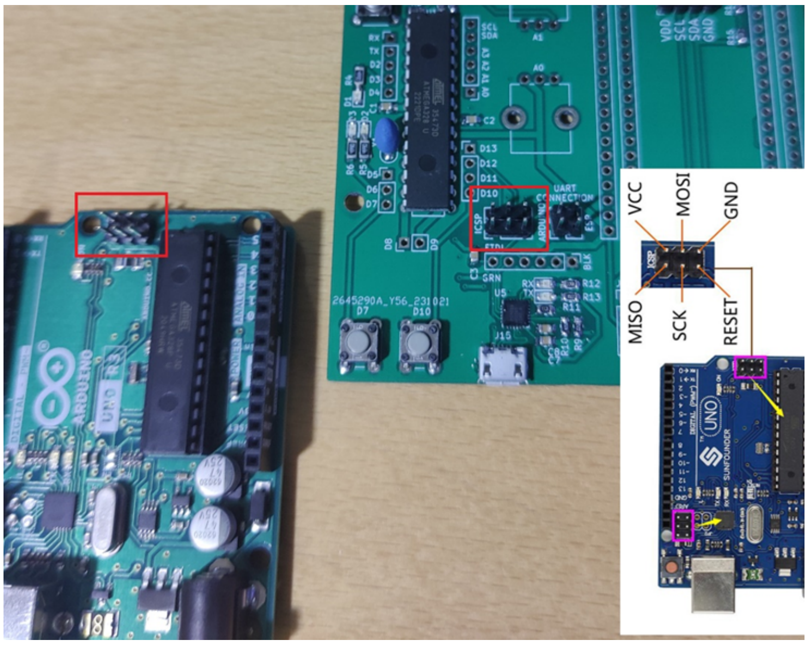
- ATmega328
- ATmega328P (5V, 16 MHz)
- ATmega328P (3.3V, 8 MHz)
- ATmega168 (5V, 16 MHz)
- ATmega168 (3.3V, 8 MHz)

o Select the ‘Arduino Pro or Pro mini’ under ‘Boards

o Select the newly made ‘ATmega328’ under Processor

o The serial port should be the Arduino that is programmed with the ‘ArduinoISP’ sketch from the top of this guide

o Next up is the connectors. The Arduino have an ‘ICSP’ header for initial programming which we will use.



o Connect all pins EXCEPT the Reset pin to each other. The connectors are oriented the same way so it’s straight forward.

o The Reset pin should be connected to PIN10 on the Arduino – we need this to be able to reset the board from the Arduino.

o Now you can press the 'Burn Bootloader' in the Arduino IDE and wait a few seconds while the initial program gets transferred.

o The board is now done and ready to use. Unplug from the ICSP and connect directly through the Micro-USB connector on the board.

o NOTE: change the Processor in the Arduino IDE to 'ATmega328P (5V, 16MHz)' in order to program the board.

§ This is a technicality since the bootloader is way more picky with the processor type while the Arduino IDE is much more lenient. As long as the bootloader can communicate with the system all is good.

Enclosure

Super easy – just print out the number of enclosures you need.

Link to enclosure for the platform:

https://cad.onshape.com/documents/849b2bd61cf74129f25efaf9/w/3d098ec1a1e2c88287bac234/e/6880bab1416d5353b95d6e91?renderMode=0&uiState=65576d122bf418313486a33d

Class 1 -structure

Class 1 – Introduction, sensor technology in general, signals, programming and protocols.

- Welcome and introduction (10 min)
- What is a sensor (5 min)
- Input/algorithm/output model of a system (5 min)
- Signals (10 min)
    - Digital
    - Analog
    - Communication
- Bits, bytes and the binary system
    - Hexadecimal numbers
- The microcontroller – the brain of the system (10 min)
    - What is a controller
        - Comparison with a computer
        - HW/FW/SW
- Break (10 min)
- Pull up/down and the importance of knowing the state of pins (5 min)
- Programing tasks (15 min)
    - Blink
    - Button to turn on LED (maybe reverse logic using !)
    - (Button as toggle (Add variable))
- Blink without delay – we only have one core! (5 min)
- Programming tasks (15 min)
    - Blink without delay, make it yourself

    - Reading from analog inputs
    - Writing to analog outputs (PWM)
- Summary of digital/analog write/read (5 min)
- Break (10 min)
- Serial port and serial terminal (10 min)
- Programming tasks (20 min)
    - (Send button presses to serial terminal)
    - Send analog values to serial terminal
- IF THERE IS TIME: More on the serial port (15 min)
    - Connection to computer programs
    - Protocols
        - Unify the communication

Class 1 -notes

Introduction, sensor technology in general, signals, programming and protocols.

This document contains detailed notes for each topic that was discussed during the class.

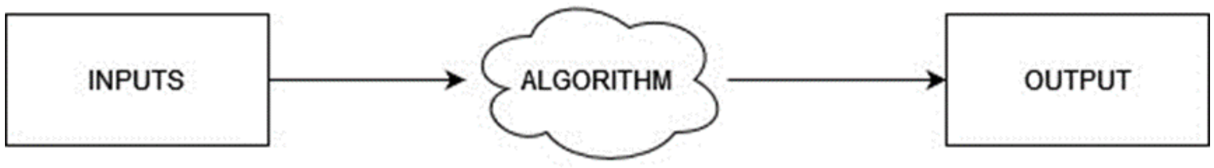
Now, a detailed look for each

What is a sensor

A device that can measure something

Input/algorithm/output model of a system

All systems can be divided into a smaller system consisting of 3 elements: input, algorithm, output



We have a lot of different sensors that can measure a lot of different things in our environment:

Temperature, Humidity, pH, pressure, Frequency, light, sound and so on. Feel free to expand the list yourself.

Similarly the output section contains actuators for the similar inputs: motor, speaker, heating elements / cooling, magnets and so on.

The last element is the ‘algorithm’ which is the main contents of the classes. The algorithm is how the system controls the output, based on the input sensor values.

Signals

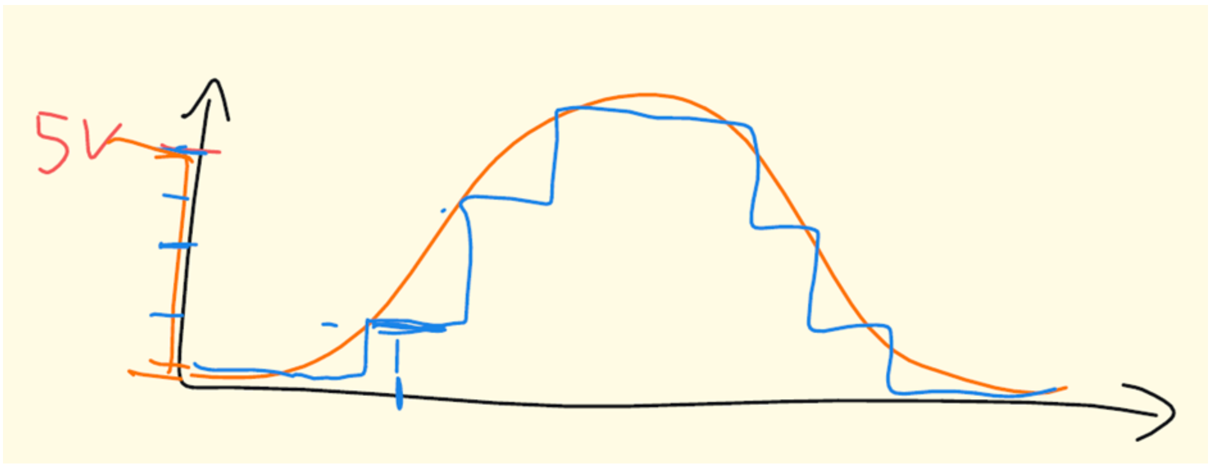
A signal can be divided into 2 categories – digital and analog.

The digital signal can have 2 values: 0 and 1.

The analog signal can take infinitely many numbers and we have infinite resolution on the signals. We can always 'zoom in' a bit further where the digital signal has a finite number of positions.

The problem with digital programming is that we always have to convert our analog signals into digital signals. This creates a bit of noise or 'translation error'.

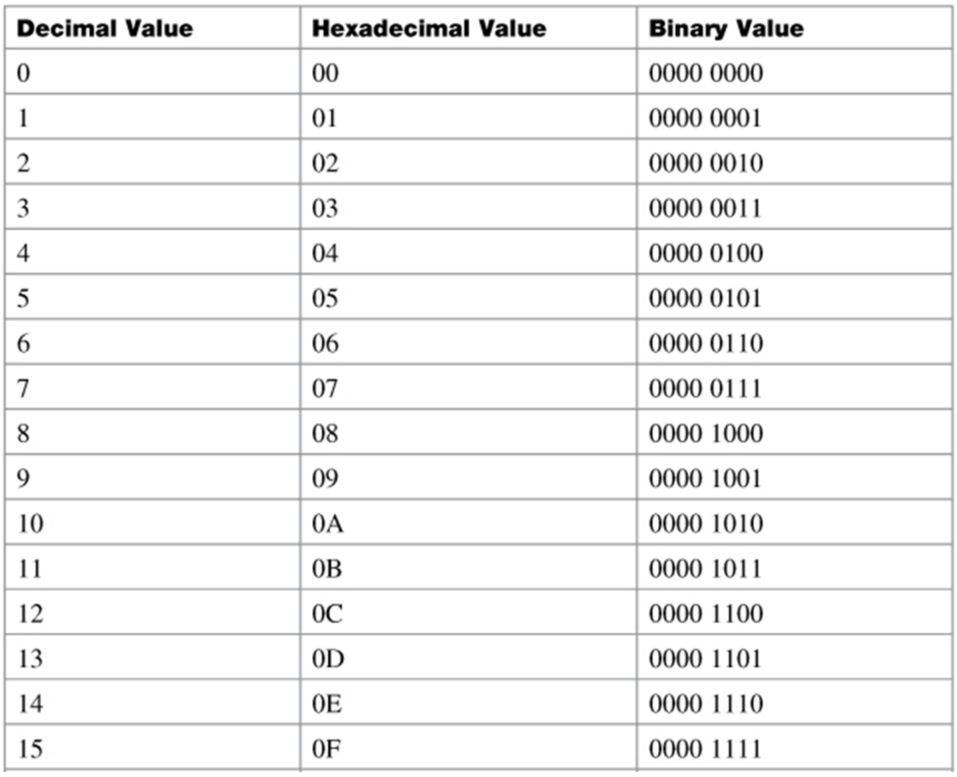
Take the following picture as an example



The analog (red) signal is smooth but the digital representation (blue) is stair-shaped because of a lack of resolution during the translation.

Bits, bytes and the binary system

We’re talking about how many bits of resolution we have which leads us to a brief discussion on how different number systems work. Here we’re using binary and hexadecimal numbers:



Decimal Value	Hexadecimal Value	Binary Value
0	00	0000 0000
1	01	0000 0001
2	02	0000 0010
3	03	0000 0011
4	04	0000 0100
5	05	0000 0101
6	06	0000 0110
7	07	0000 0111
8	08	0000 1000
9	09	0000 1001
10	0A	0000 1010
11	0B	0000 1011
12	0C	0000 1100
13	0D	0000 1101
14	0E	0000 1110
15	0F	0000 1111

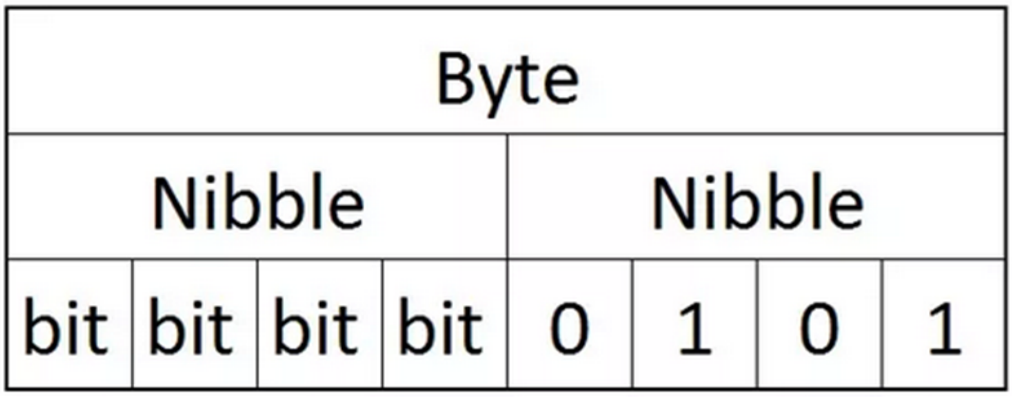
So in base-10 we will have ‘ones’, and ‘hundreds’ where in the binary we will have ‘ones’, ‘twos’, ‘fours’ and ‘eights’.

We can write binary numbers very short by converting them to base-16. Here we have ‘ones’ and ‘sixteens’ and so on.

- A single binary number is called a bit
- 8 bits make a byte
- 4 bits make a nibble

Hexadecimal numbers

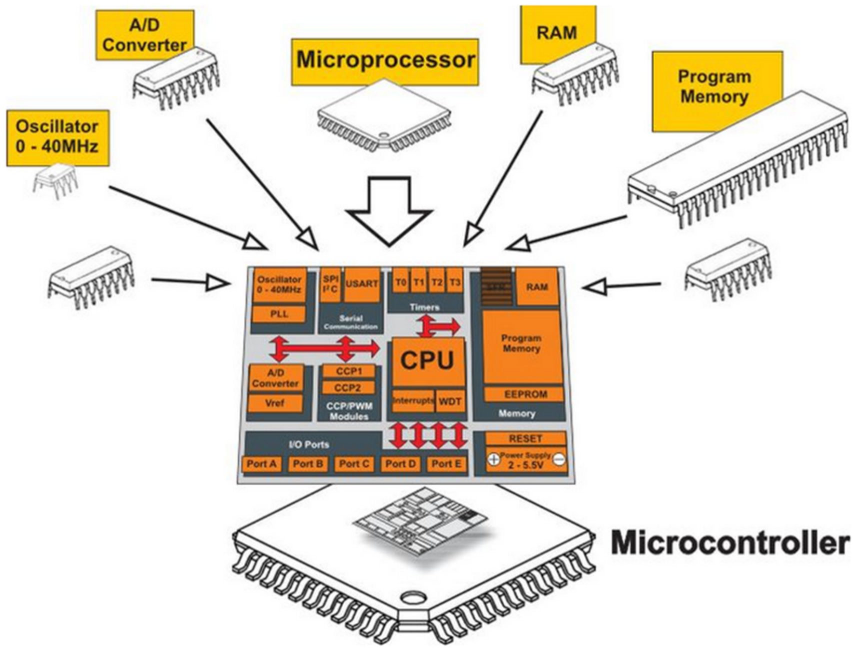
Each nibble can be written using a single hexadecimal value, creating easy-to-read bytes since they will have 2 hexadecimal values. For example 0xAF (where 0x is used to denote that a hexadecimal value is being described) AF is the same as ‘175’ in base-10.



Byte							
Nibble				Nibble			
bit	bit	bit	bit	0	1	0	1

The microcontroller – the brain of the system (10 min)

The main difference is that all the different parts in a computer are combined into a single package that we can simply place into our design with very few extra components.



Microcontroller Internal Parts

HW/FW/SW

When talking about hardware we can apply the rule 'does it hurt when I throw it at you'.

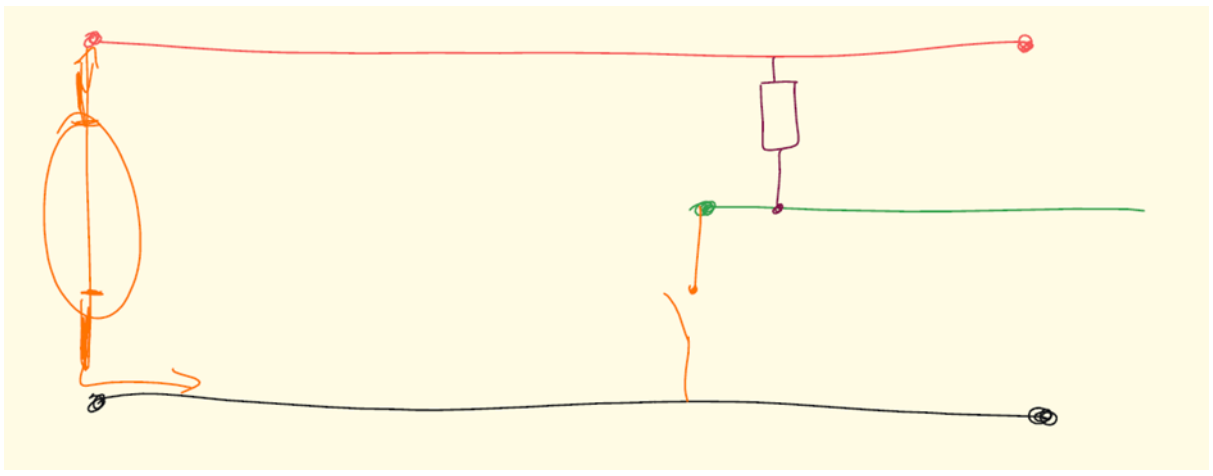
Hardware will hurt.

Software exists only inside a computer

Firmware is software written for a specific hardware platform which means that if we write a program for an Arduino, we will have a piece of firmware.

Pull up/down and the importance of knowing the state of pins

It is very important to keep all inputs in a well-known state all the time. Sometimes it can become very critical to have a known state, for example if the system controls valves that can lead to flooding or other serious issues.



In the picture above a small resistor has been added to the input pin (green). When the switch (orange) is open, the pullup will pull the input pin up to the supply voltage (red). When the switch is closed, the input pin will have a low-resistance path to ground, pulling the pin low.

If the resistor was NOT there, we would create a short circuit between the supply voltage and ground when pressing the button which is NOT desirable.

Blink

The blink example can be found in its entirety under the built-in examples in the Arduino IDE.

Button to turn on LED (maybe reverse logic using !)

We can write this a single line:

digitalWrite(3,digitalRead(7));

Since 'digitalRead(7)' will return the value of the pin-state we can use that directly as the value for the output pin

However, since we have a pull-up enabled we will see the LED being on when the button is not pressed. We can change this behavior by adding a '!' to the value:

digitalWrite(3,!digitalRead(7));

The '!' means 'not' so the result is that the signal is inverted. This is a quick way to change the logic of a program statement without changing a whole lot.

Blink without delay – we only have one core!

On a computer we can just open up a list of programs and have them all run but on a microcontroller it is slightly different. We only have one core and can only do one thing at a time. Therefore if we use 'delay' we can ONLY spend our time delaying. That means that we're going to miss out on sensor inputs and changes if we don't sample them quickly enough.

The 'blink without delay' example is available as a built-in example under examples -> 02. Digital -> blink without delay. It is very well documented.

Reading from analog inputs

Until now we have talked about digital inputs but we also have the option to sample analog inputs and get a value from them that is more than 0 or 1.
The analog inputs have 10-bit resolution giving us values from 0 to 1023.

Using the function 'analogRead(*pin*)' we can get the value from the pin.

Writing to analog outputs (PWM)

Similarly we also have analogWrite(*pin*, *value*). Although we only have 8-bit as output so we can only set 256 levels.

Summary of digital/analog write/read

See the Arduino language reference for more information about the specific functions. Available here:

https://www.arduino.cc/reference/en/

Serial port and serial terminal

Having a microcontroller is no fun when you cannot show data anywhere. Luckily we can send data off to a computer (or something else) with the help of the serial port!

Initialize the serial object by writing

Serial.begin(115200);

Where '115200' is the number of characters per second you want to write. This number is from a list of standard values.

We write to the serial port by using:

Serial.println(whatToSend);

The println will add a line change after each print so it is easier to read. Replace 'whatToSend' with the information you want to send. That could be a variable or a string encapsulated in "".

Open up the Serial terminal in the Arduino IDE by pressing the small looking glass in the upper right corner. Remember to set the baud-rate according to the number you set in the code.

Enjoy your data on the screen!!

You can also open the Serial Plotter from the 'Tools' menu in the IDE.

Class 2- structure

Class 2 – Wifi, Signal conditioning, Windows forms programming (C#), how to use electronic modules as building blocks.

- Installing the ESP32-Arduino section
- Connecting to the internet using WiFi (10 min)
- Storing data online (10 min)
    - Data in 'the cloud'
    - Database systems (briefly)
- Code examples with WiFi (30 min)
    - Retrieve a website
    - Show a web server
    - Send data to a database system
- Break (10 min)
- Software for datalogging (15 min)
    - C# for windows programming (Visual studio community)
- Making our own program (15 min)
    - Create a winforms program with the serial port functionality
    - Receive data and save it to a file
- Creating a protocol of our own (20 min)
    - Sending data back
    - Controlling the controller (from our computer)
- Break (10 min)
- Other communication systems (10 min)
    - I2C, SPI, RS-485, CAN
- Signal conditioning (15 min)

- Turning our signal into something we can understand
  - Digital/analog/communication
  - Lowering /raising voltage
  - Adding ‘glue’ logic to the signal

- Outputs (25 min)
  - Monitoring only
  - Actuators and other outputs
  - Acting on our sensor input – now we have a control system!
  - Controlling outputs
    - PWM (as the analogWrite)
  - Output stages
    - Power control
      - MOSFETS
      - Relays
      - H-Bridges

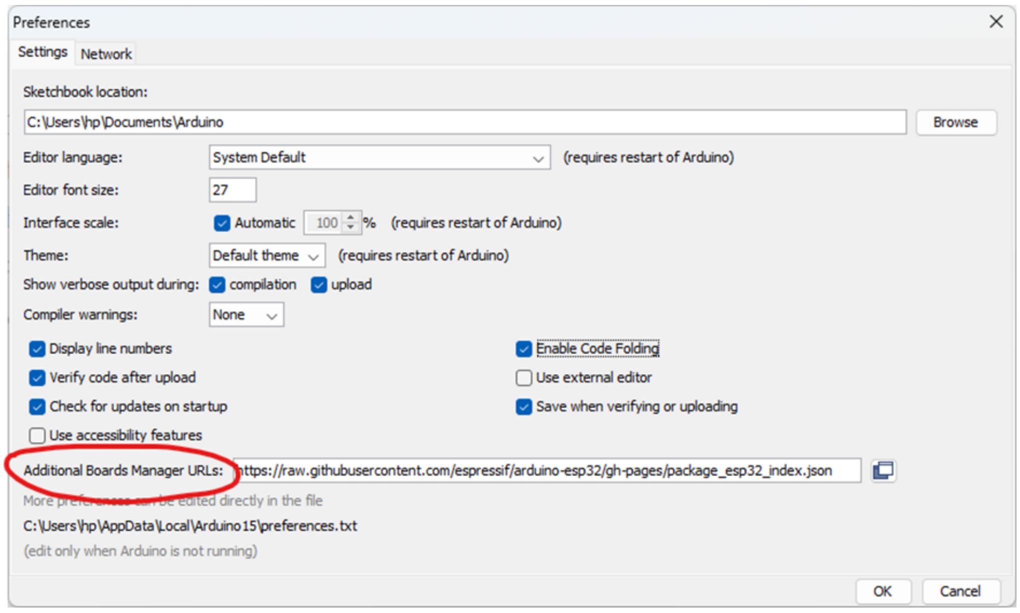
Class 2 -notes

Class 2 – Wifi, Signal conditioning, Windows forms programming (C#), how to use electronic modules as building blocks

Installing the ESP32-Arduino section

In order to be able to use the ESP32 module in Arduino, we need to install the framework for it.

This is done by opening the preferences (File -> Preferences) and adding the following line to the 'Additional board manager URLs' list:



Sketchbook location: C:\Users\hp\Documents\Arduino

Editor language: System Default (requires restart of Arduino)

Editor font size: 27

Interface scale: ✓ Automatic 100 % (requires restart of Arduino)

Theme: Default theme (requires restart of Arduino)

Show verbose output during: ✓ compilation ✓ upload

Compiler warnings: None

✓ Display line numbers
✓ Enable Code Folding
✓ Verify code after upload
☐ Use external editor
✓ Check for updates on startup
✓ Save when verifying or uploading
☐ Use accessibility features

Additional Boards Manager URLs: https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json

More preferences can be edited directly in the file

C:\Users\hp\AppData\Local\Arduino15\preferences.txt

(edit only when Arduino is not running)

The line you want to add is:

https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json

Following that you need to go to Tools -> Board -> Board manager and search for 'ESP32'. There will be at least 2 results. Select the 'esp32' by Espressif systems and install that.

For further instructions, have a look at this tutorial: https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/

This is a one-time setup and should only be done if you install the Arduino IDE on a new computer.

Connecting to the internet using WiFi

This will briefly showcase a few of the examples that follow the ESP32 installation into the IDE.

Firstly, we need to select the right board to enable the examples. Go to Tools->Board->ESP32 Arduino and select the ‘ESP32-WROOM-DA Module’.
The list is quite long and a lot of the modules might work, but have small differences between them.

Now, go to File->Examples and you will see a list of examples for this board. First in the list is the built-in examples and examples will work with any platform. The last section is the board-specific examples.

We’re going to focus on two examples in particular: the ‘HTTPClient’ and the ‘WebServer->AdvancedWebServer’ examples.

Common for both of them is that they require a WiFi to connect to.

Replace ‘SSID’ with the network name and ‘password’ with the password for the network.

Next, upload the example – there should not be need for further changes

When the IDE writes ‘Connecting….’ Press and hold the ‘BOOT’ button on the module.
This will allow the module to enter bootloader/programming mode which is required in order to program the board.

The bootloader is a small program that is stored permanently in the module and is only accessible if the boot button is pressed during a reset. This reset comes automatically through the programming procedure – you just need to hold the boot button until the upload begins. When it is running you can let go.

After the programming procedure is done, you can press ‘EN’ (enable/ reset) and the module will run the code you programmed it to do.

For the HTTPClient example, the code will try to access a website and display the HTML contents on the serial terminal.

The AdvancedWebServer example will provide you with the IP address of the server through the terminal. Copy the address and open it with your browser. This should display a webpage that is hosted on the module itself! Pretty neat!

Storing data online

This will only be covered briefly – You are more than welcome to ask for more information on this topic if needed.

We can set up a webserver online (preferably on a domain we can access) with a database system. For testing purposes we can install XAMPP which contains all we need in one simple package – go to https://www.apachefriends.org/ and find the installer for you. This will install the following: Apache web server, PHP (needed for server-side coding) and a MySQL database system. This will also install 'phpMyAdmin' which is a database administrator application that runs on the web server. This interface is used to ease the creation and management of database tables on the system.
Please see the files section for an example on this cloud solution (Cloud website.rar).

Unpack the contents and place the contents of the 'cms' folder into your webroot. The .sql file needs to be inserted into the database which can be done using the PhpMyAmin interface.

You will need to create an entry in the 'users' table which can be done using the 'insert' tab in PhpMyAdmin. Remember to set the 'MD5' property on the password field since we NEVER store passwords in clear text in a database. Ever.

Data in 'the cloud'

Having a webserver is wonderful but where is it hosted? Is it our own? Could someone else shut it down if they deem the services too expensive? These are all concerns that we need to take into account when offering data to a 'cloud service'. Who owns data and who owns the systems that collect them.

Data online and in cloud services are not inherently bad but we need to think about the pros and cons when sharing data.

Send data to a database system

Looking a bit more closely at the database system presented above we would need to adjust the HTTPClient example from before to allow uploading data to our system.

We will leverage the way we request web pages with our browsers, specifically the GET request.

Take this website URL: https://example.com/test.html?argument=value&arg2=719

Take a close look at what happens after the '.html'. The first sign is '?' and then after that, we have a set of arguments with corresponding values attached to them. In this specific case we have two arguments called 'argument' and 'arg2'. This is the name of the arguments, with the values 'value' and '719' respectively.

Pointing to our own webserver and to the 'catch.php' page, we can utilize this.

First we need to have a unique identifier for our sensor so the database can sort the sensors from each other. Log in to the page and create a sensor. This will give you an API key. This needs to be supplied as the argument 'API=YOUR_KEY_HERE'. Following that, we have some data. This can take any form in order to make the system more versatile: 'data=YOUR_DATA_HERE'.

Combining this, we need to request the following page:

https://YOUR_DOMAIN/catch.php?API=YOUR_API_KEY&data=YOUR_DATA_HERE

where you substitute the domain for your web address (or IP for local applications), 'YOUR_API_KEY' for the API key you created in the system and 'YOUR_DATA_HERE' with the actual data you want to save.

The request should come from the ESP32 module – so the data we send can be sensordata directly sampled from our sensors.

C# for windows programming (Visual studio community)

C# is a great language for creating native windows applications. For developing we can use the freely available tool 'Visual Studio Community', available from this page: https://visualstudio.microsoft.com/vs/community/

You will need to download an installer and install the '.NET desktop development' in order to be able to make your own programs.

When installed, create a new WinForms (.NET) program.

First time you start the program you need to open the toolbox. (View->toolbox). Pin it to the left side of the window with the pin in the upper right corner of the toolbox window.

After this, you are freely able to create any program you want. Try to play around with different controls and see what they do.

Making our own program

Adding controls to the GUI is as simple as dragging and dropping the controls needed onto our window.

Double clicking a control will create the associated click event handler code and bring us there so we can write the code associated with the click event.

There are many resources available online for further exploration on the controls and how they are used. Also they will be much better than the notes written here.

Create a winforms program with the serial port functionality

One control with a certain interest is the 'Serial port'. By having one of these in the program we can easily communicate with our controller over the standard serial interface.

Remember that we cannot have the same port open multiple places – so we cannot have the port open in our C# program while programming through the Arduino IDE.

Limiting the amount of data we send – Creating a protocol

As can be seen in the Arduino code attached in this class, we have added a simple control mechanism to limit the amount of data being sent back and forth.

This is an important step to take since it will leave more time on the controller to.. well, control. Handle input and outputs. Sending data over the serial port is quick but it takes _some_ time. This means that by limiting the number of data we send we can increase the speed of the main loop and thus be more responsive.

Creating a protocol of our own

The simplest way of limiting the data is to wait for a special character. In this case we chose 'A'. First we wait for something to come in at the serial port. When something does arrive, we check if it matches our special character. If it does, we reply by returning the sensor data. If the character does NOT match we don't do anything.

```
if(Serial.available()>0)
```

```
{
        int rec = Serial.read();
        if(rec=='A')
        {
        //response code here
        }
}
```

Sending data back

The only thing needed is the response code that we can choose to be what we want. The simplest way to return data will be to return the value of the input port we chose. As long as we end with a newline, we can add more data to our response.

Controlling the controller (from our computer)

Until now we have just sent a single character ‘A’ to request data from our platform.

There is nothing to limit which character we send and having a set of buttons triggering different characters to be sent will allow us to respond differently based on what request we get.

That will now allow us to control the Arduino platform from the computer side, greatly increasing the versatility of the system.

Class 3 – structure

Building blocks and schematics

- How to build our own system

o Building blocks (10 min)

o Schematic capture (20 min)

o Footprint linking (10 min)

o Layout and routing (30 min)

§ Physical constraints

§ Parts in fixed positions (connectors, buttons)

§ Route!

o Manufacturing (20 min)

§ Etching

§ Milling

§ Ordering

Class 3 – notes

Modules

Modules are building blocks for electronics. Each system can be broken down into these blocks. This is a (non-exhaustive) list of modules that can be used to create a system.

Power supply

- Voltage regulator (LDO)
- Switching regulator
- Rectifier (AC-DC)
- Reverse polarity protection

Signal conditioning

- Voltage divider
- Microcontroller
- Digital input / digital output
- Analog input / analog output
- Pull-up / Pull-down
- Logic
- Op-amps

Lights and displays

- LED, simple (and 4-pin RGB bulbs) (Charlie-plexing?)
- LED, Addressable (WS2812 style)
- OLED display

Power control

- MOSFET (N and P)
- Relays
- Motor driver (H-Bridge)
- Multiplexers, shift registers

Signal conditioning

Converting a signal from one type to another is one of the most important things we have to do while building a sensor system. In order to use all the resolution on the AD-converter we might need to amplify a signal or, if the signal is too high, lower it so it will fit in the range of the converter

Turning our signal into something we can understand

If you have to remember just a single thing then remember this: **Controllers only understand voltage**

We might have a sensor that changes in resistance as a result of the variable it is measuring. A light-dependent resistor or a flex-sensor. These resistances have to be converted to a voltage.

It could also be a current that we need to measure – again we need to convert it into a voltage to be able to read it with the AD-converter.

Digital/analog/communication

We have 3 types of inputs from sensors

Digital – they will give a high/low signal depending on what they are sensing. A variant of these will only pull an input towards ground – meaning that you have to provide the pull-up resistor for the input. Like a button for example. When it is open it's not 'helping' the input pin in any direction and we will have to provide a way for the pin to be in a well-defined state.

Analog – We will get a voltage from the sensor, following a predetermined formula – look in the datasheet for the sensor to see how the output relates to the sensing. This analog voltage we want to measure with our AD-converter. Then we can use our programming to determine the value of the sensor and use that to further control our system

Digital protocols – some sensors have a digital interface. It can be on an I^2C bus or another communication interface. These sensors usually can be queried for a reading and will answer over the digital interface.

The main takeaway from these sensors is that the measurement and signal conditioning is usually done on the sensor itself so we just need to ask for a sample and then we will have a number to directly work with afterwards.

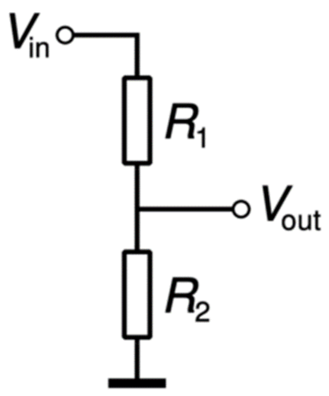
These sensors are nice to work with since they are easy to interface and usually have fewer wires than other sensors. However, we need to make sure that we read the datasheet carefully in order to understand what values we get from the sensor.

Lowering /raising voltage

A large part of signal conditioning is the adjustment of the voltage range. For this we have 2 types of systems.

Most common is the voltage divider

The construction of the voltage divider looks like this



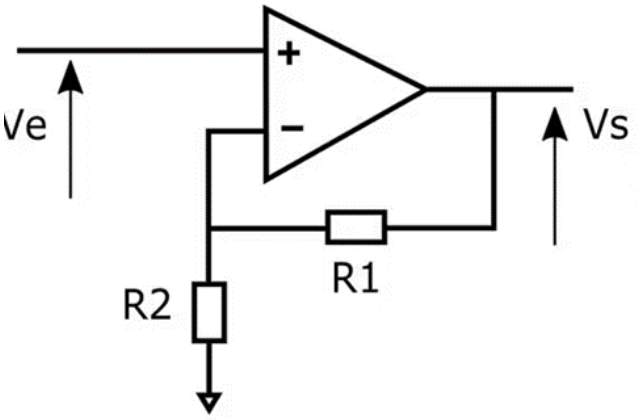
The output voltage is determined by the ratio of the resistors R1 and R2.

You can read much more about the voltage divider here: https://learn.sparkfun.com/tutorials/voltage-dividers/all

Also, you can see the formula (it is quite simple) and even find a calculator for the divider on the page.

A quick note: if you make R1 and R2 the same value, the ratio between Vin and Vout will be ½, meaning that you will always get half the voltage out.

OP-amps



Operational amplifiers are what they say; amplifiers.

The symbol looks like this, the distinct triangular shape with a positive and negative input.

The premise for the workings of an op-amp is that ‘no current enters the inputs’ meaning that the load imposed on the thing we are measuring is 0.

When 0 current is flowing into the amplifier we only have voltages to deal with which makes the formulas much simpler.

The op-amp is a very versatile component where different support components around it will allow for almost all types of mathematics in the input signals: multiply, divide, summation, integration and differentiation. The op-amp can also be used as a comparator where an input signal can be compared to a reference signal.

Much more on how to set up the op-amp circuit is available here:
https://www.arrow.com/en/research-and-events/articles/fundamentals-of-op-amp-circuits

Warning: the op-amp can require a both positive and negative supply voltage. Keep this in mind – you might need to have both, depending on your signal type!

Use the light sensitive resistor as an example

This section will refer to the nice write-up found on the sparkfun page of the voltage divider:

https://learn.sparkfun.com/tutorials/voltage-dividers

Go to the section ‘Reading resistive sensors’

Using this approach we can use a voltage divider to convert all resistive sensors into a voltage signal that we can sample using our AD-converter

Outputs

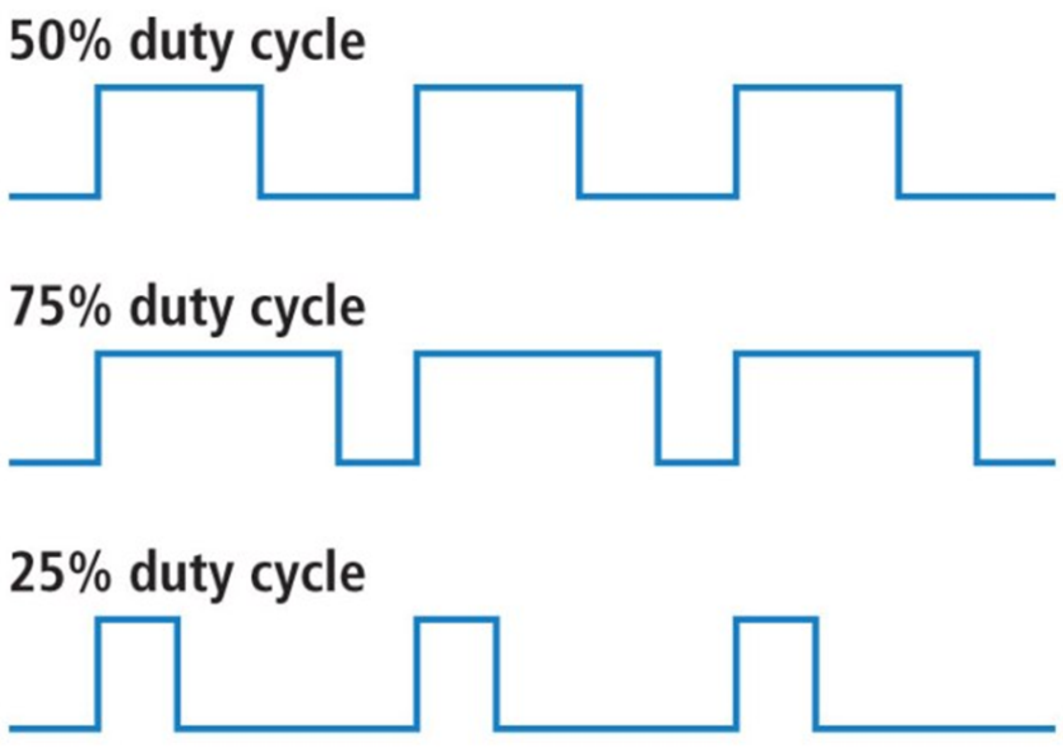
Without any output our system is limited to monitoring of signals only. We can save data but as long as we don’t control anything we can only record the sensors.

PWM

Pulse-Width-Modulation is a great way to control outputs digitally but make them behave almost like analog signals

There is a great article about PWM here:
https://learn.sparkfun.com/tutorials/pulse-width-modulation

The most important is that we can control the amount of 'on time' in the signal.



This is basically turning on and off the signal very fast. This (for an LED) will look like we are dimming the light. For a motor, we will be able to control the speed of the motor.

MOSFETS

A MOSFET is a variant of a transistor which is a digital switch. A small signal from our controller can turn on the switch that holds back a large current.

A MOSFET is like a relay but it has no moving parts meaning that it can switch on and off much faster than a mechanical design. If we supply the MOSFET with a PWM signal we can control large things such as pumps or motors (or large LEDS) that require high voltages to work.

An entry on how the MOSFETS work can be found in the schematic building blocks:
https://github.com/HansPeterHaastrup/schematicModules/blob/main/powerControl/MOSFET/MOSFET.md

Kicad design walkthrough

KiCAD is a free and powerful tool for Printed circuit board (PCB) designs. While I did a very quick intro to the workflow: Schematic capture -> footprint allocation -> layout -> ordering -> assembly, the following video series (10 videos) with 10-20 minute videos covers all the steps, including how to create your own components along with other important aspects of PCB design.

https://www.youtube.com/watch?v=vaCVh2SAZY4&list=PL3bNyZYHcRSUhUXUt51W6nKvxx2ORvUQB

The video series is based on KiCAD version 4.07 (we're version 7 now) but it still contains loads of valuable information and the process is very similar. Just be aware that certain shortcuts might have changed and minor changes to the workflow might occur. It's not a great change though.

Building blocks

So how exactly do we just build the schematic – from thin air?

In my world electronics are like building blocks. You take this block here and combine it with this other block there. By using this approach we quickly can build up the complete circuit we need.

I have started making a library of these building blocks along with notes, available here: https://github.com/HansPeterHaastrup/schematicModules

Manufacturing

Making the PCB we just designed is quite an important part of the process. We need something physical to mount our components to!

We have two options – make it ourselves or have somebody make it for us.

The DIY approach is appealing, but it will take some time and knowledge to get working properly. Also it will require some special tools to get started.

Ordering online is (as with many other things) a cheaper and faster way. We just have to wait instead.

I usually order from JLCPCB.com (china) and have boards delivered within a week. This is time used to order components and other things needed for the project at hand.
 Other manufacturers exist but usually the Chinese ones are the cheapest which makes sense for prototyping work.

A practical example with a sensor