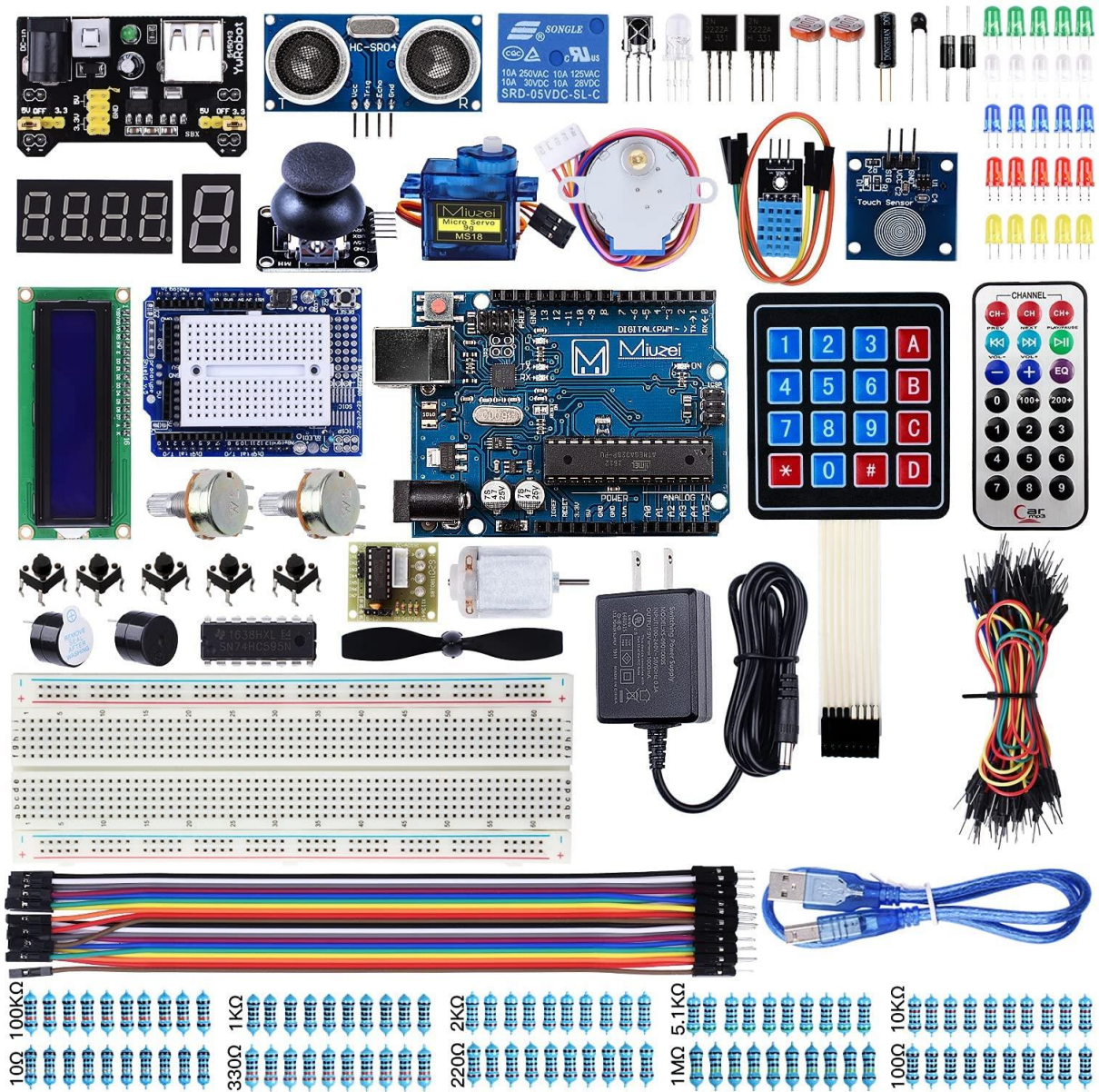


## Advanced



## Version table

Version	Date	Description
1.0	20-11-2021	Initial version after review
1.1	14-12-2021	Processed additional review comments
1.2	10-01-2022	Added properties link

## Contents

1	Introduction .....	4
1.1	Showing the different parts of the learning outcome multiple times .....	4
1.2	Demand based approach .....	4
2	Arduino board and breadboard .....	5
2.1	Arduino board .....	5
2.2	Challenges .....	5
2.2.1	Arduino pinout challenge .....	5
2.3	The use of breadboard and components.....	5
2.4	TinkerCad.....	5
2.5	Challenges .....	5
2.5.1	TinkerCad circuit challenge .....	5
2.5.2	Breadboard circuit challenge .....	6
2.5.3	Arduino self-test challenge .....	6
3	Basic embedded software structure (Arduino framework, no OS) .....	7
3.1	Program flow and timing, libraries.....	7
3.2	Challenges .....	7
3.2.1	Blinking LEDs challenge.....	7
3.2.2	Button debounce with millis challenge .....	7
3.2.3	Button function challenge .....	7
3.3	Programming with states .....	7
3.3.1	State machine expert challenge .....	8
4	Basic I/O techniques .....	9
4.1	Challenges .....	9
4.1.1	I/O techniques challenge .....	9
4.1.2	I/O techniques Kahoot challenge.....	9
5	Sensors.....	10
5.1	Sensor research, connecting sensors, reading sensor data, processing sensor data. Analog input.....	10

5.2	Challenges .....	10
5.2.1	Analog sensor and Serial Plotter challenge .....	10
5.2.2	LDR sensor challenge .....	10
5.2.3	Scaling challenge .....	11
5.2.4	Running average challenge .....	11
5.2.5	Hysteresis challenge .....	11
6	Libraries.....	12
6.1	Challenges .....	12
6.1.1	DHT Library challenge .....	12
6.1.2	Ping sensor library challenge .....	12
7	Object Oriented Programming (OOP).....	13
7.1.1	Introduction Objects en Classes: OOP .....	13
7.1.2	Traffic light challenge.....	13
7.1.3	Bank account challenge .....	14
7.1.4	LED class expert challenge .....	15
7.1.5	Button class expert challenge .....	15
7.1.6	Timer class expert challenge.....	15
7.1.7	Traffic light expert challenge .....	15
7.1.8	Car system expert challenge.....	15
8	Actuators.....	16
8.1	Pulse width modulation (PWM).....	16
8.2	Challenges .....	16
8.2.1	PWM LED challenge .....	16
8.2.2	Servo control challenge .....	16
9	Communication.....	18
9.1	Challenges .....	18
9.1.1	Remote control challenge.....	18
9.1.2	controlled testing expert challenge .....	19
9.1.3	Arduino 2 Arduino serial communication expert challenge.....	19
9.1.4	Arduino 2 Arduino I2C communication expert challenge .....	19
10	Project.....	20

# 1 Introduction



This work document will help you reach the learning outcomes for Technology in the 'Startsemester verdieping'. It contains online sources and challenges you can use to practice different parts of the learning outcome.

There are also 'expert challenges'. These cover topics which are strictly not part of the learning outcome, but certainly will better prepare you for coming semesters!

## 1.1 Showing the different parts of the learning outcome multiple times

Besides the challenges you will also work in a project to apply your skills in a bigger context. This way you are able to demonstrate the different parts of the learning outcome multiple times, which is required to show that you've mastered the learning outcome.

## 1.2 Demand based approach

Because the course is demand based, you are free to choose how you are going to use this document. **However, if you are skipping challenges in this document, please do not skip the underlying part of the learning outcome. This means you will have to come up with an alternative way of showing that part.**

## 2 Arduino board and breadboard

### 2.1 Arduino board

See <https://pdfs.semanticscholar.org/5361/0d6c9ec45b1ef2f4cd0d9c241d8462e39db4.pdf> for a comprehensive introduction on the Arduino board.

### 2.2 Challenges

#### 2.2.1 Arduino pinout challenge

Find a comprehensive picture of the Arduino Uno board layout which you can use for your future projects.

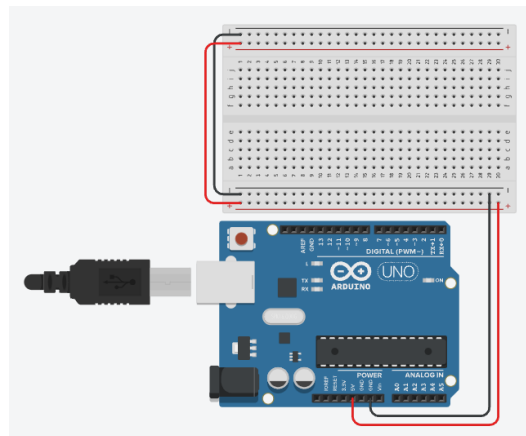
- Does it include the functions of all pins with pin numbers?
- On this picture, can you see which pins are capable of doing PWM output?
- Which digital I/O pins had you better not use and why not?

**Tip:** You could place this picture on your desktop as a quick reference!

### 2.3 The use of breadboard and components.

In the 'verdieping' we do not work with the RICH Shield anymore, but with a breadboard and components. See <https://startingelectronics.org/beginners/start-electronics-now/tut1-breadboard-circuits/> for the use of a breadboard and components.

### 2.4 TinkerCad



TinkerCad is an online tool which can be used to create Arduino circuits on a breadboard and simulate them. The main steps you have to take:

1. Go to <https://www.tinkercad.com/> and create an account and sign in.
2. Choose the 'Circuits' section to go to create or edit a circuit.
3. Find a place the Arduino with breadboard in the working area (search for 'arduino uno').
4. Find and place the desired components and connect them in the proper way to the breadboard.
5. Select Code->Text to write C code.
6. Click 'Start Simulation' to start the simulation.

TinkerCad can also be used to just create a design and use it for documentation.

### 2.5 Challenges

#### 2.5.1 TinkerCad circuit challenge

Use TinkerCad <https://www.tinkercad.com/> to create the following circuits:

- LED which can be controlled with a digital output pin.
- Button which can be read with a digital input pin.
- Potentiometer which can be read using an analog input

- LED which can be faded with a PWM output.

And of course, write code in TinkerCad to test your circuits!

### 2.5.2 Breadboard circuit challenge

Build up the same circuits as above on a real breadboard as above and test them. Do you see any difference?

### 2.5.3 Arduino self-test challenge

How nice would it be to be able to quickly test your Arduino? At least you want to know if all digital inputs, digital outputs and all analog inputs are working.

Note that in this challenge there are no simple right or wrong answers. Try to think of an efficient way with not too many steps for the tester.

Hint 1: Make use of serial print to give instructions to the tester!

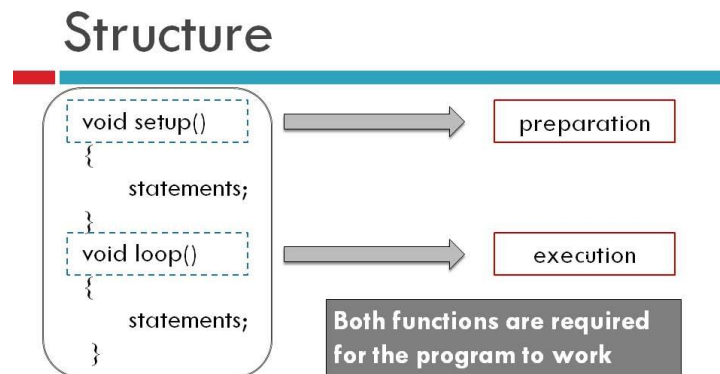
Hint 2: Make use of serial read to let the tester proceed to the next test step or to skip a test.

- Which external components (like buttons, LEDs) would you use and how many of them do you need?
- Write a program to test all digital I/O pins and analog pins. Use your loop programming and function skills to make it an efficient program.
- What would your test approach be if you have only one button, one LED and one potentiometer?

Hint: In this case the tester would have to take more steps but that's ok.

**Tip:** If your self-test program is working, save it so you can use it whenever you are in doubt if your Arduino is ok or not.

### 3 Basic embedded software structure (Arduino framework, no OS)



#### 3.1 Program flow and timing, libraries.

See <https://www.youtube.com/watch?v=lyxY1uQyY9U> on the program flow of a standard Arduino program and the use of delay().

See <https://www.youtube.com/watch?v=hD3cR25MbW8> on the use of millis().

#### 3.2 Challenges

##### 3.2.1 Blinking LEDs challenge

- Using millis(), write a program that blinks one LED x times each second and another LED y times each second. x and y can have any value > 0.

Notes:

- Do not put all code in the main loop but create your own timer functions. This way your code will be much more readable and reusable!
- You could for example create a StartTimer() and a CheckTimer() function which can handle 10 timers. You can use a static array to store the timer values.

##### 3.2.2 Button debounce with millis challenge

- Using the knowledge acquired about millis(), have a look at <https://www.arduino.cc/en/Tutorial/BuiltInExamples/Debounce> and test this using one button and a LED toggle.

##### 3.2.3 Button function challenge

- When this works, create your own button function so you can handle buttons easily the rest of your embedded career!
- Test this function with at least two buttons!

Hints:

- For one button you can simply create a function and also a 'lastButtonState' variable.
- When you need to handle multiple buttons you need multiple 'lastButtonState' variables. One possibility is to use the button number (starting from '0') as an index in an array which stores the 'lastButtonState' variables for each button.
- When using classes this can be handled more elegantly.

#### 3.3 Programming with states

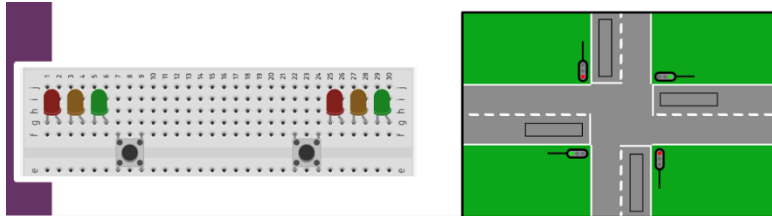
When an application has to react differently on different situations, it is useful to program these situations explicitly using states. In each state the application might act differently on the input. An example is a traffic light control application. This application might react differently on a traffic approaching sensor dependent on whether the light is green or red.



With this approach you implement a so-called (finite) state machine.

### 3.3.1 State machine expert challenge

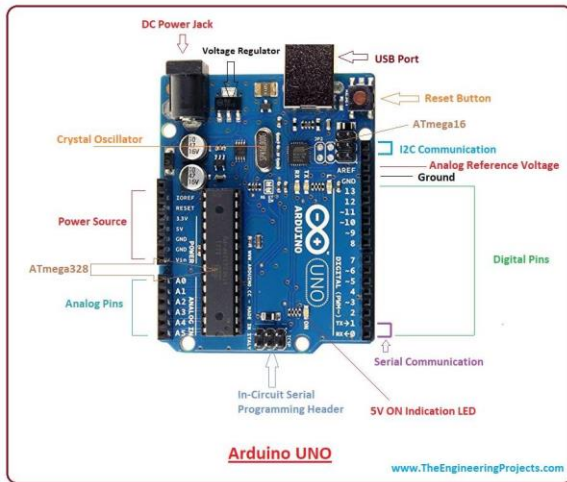
- Carefully read <https://arduinoplusplus.wordpress.com/2019/07/06/finite-state-machine-programming-basics-part-1/> and make sure you understand what is going on.
- Connect LEDs and buttons to your Arduino, representing a simple crossing with each having a traffic light and a traffic sensor. Not that the hardware is a bit simplified: one button represents two sensors on the same road. The same goes for the traffic lights.



- Create a traffic control program using state machine programming!



## 4 Basic I/O techniques



The basic I/O techniques often used include:

- Digital on/off
- Analog
- Serial UART
- I2C
- SPI

### 4.1 Challenges

#### 4.1.1 I/O techniques challenge

There are many sources explaining the different I/O techniques.

- For each of the mentioned techniques above, find a source which clearly explains its principle.
- For each of the mentioned techniques above, find some sensors or actuators using this technique.

#### 4.1.2 I/O techniques Kahoot challenge

# Kahoot!

- Create a Kahoot to check if your fellow students and your teacher know as much as you do on the different I/O techniques.
- Ask the teacher if your Kahoot can be shown in class!

## 5 Sensors

### 5.1 Sensor research, connecting sensors, reading sensor data, processing sensor data. Analog input.



At <https://fontys-fhict.auralibrary.nl/auraicx.aspx> you can see what you can borrow at the ISSD.

### 5.2 Challenges

#### 5.2.1 Analog sensor and Serial Plotter challenge

- Connect a sensor with analog output and make a program to read the analog value. A nice tool to visualize an analog signal is available in the Arduino IDE, unfortunately not in PlatformIO. See <https://arduinogetstarted.com/tutorials/arduino-serial-plotter> for an explanation on the Serial Plotter.
- Check if the values of the analog signal are as expected.

#### 5.2.2 LDR sensor challenge



At <https://arduino-lessen.nl/les/lichtgevoelige-weerstand-ldr-uitlezen-met-arduino> it is explained how to connect an LDR (Light Dependent Resistor) to an Arduino using a resistor. In this challenge you are going to read the sensor values and also determine what the best value for the resistor is.

- Connect an LDR and resistor as explained.
- Read the sensor values and see how these change when covering / uncovering the LDR. Using the Serial Plotter might be helpful.
- Vary the resistor value and see how this effects the sensor values you read.
- The resistor value is optimal when the difference between the minimum sensor value and the maximum sensor value is as large as possible.
- Use different resistors or use a potentiometer to determine the optimal resistor value.

If you are interested in how to compute the optimal resistor value and are not afraid of a little math, have a look at: <https://markusthill.github.io/electronics/choosing-a-voltage-divider-resistor-for-a-ldr/>.

### 5.2.3 Scaling challenge

When using a sensor like an LDR to dim a LED (so it is not too bright at night) you want to be able to control the amount of dimming. For this it is needed to scale the sensor input signal to a desired output signal. For mapping an input range onto an output range, you can use the Arduino `map()` function:

<https://www.arduino.cc/reference/en/language/functions/math/map/>.

- Using the LDR example above, use the `map()` function to dim a LED when it gets darker.
- Make sure the dimming lower and higher threshold (at what light level the dimming starts and at what light level the dimming ends) can be set as well as the amount of dimming.

### 5.2.4 Running average challenge

Suppose you use a light sensor to switch on an outdoor light when it gets dark. Obviously, you do not want to react on quick changes for example when a bird flies over your sensor. Therefore, you need to smooth the data before determining whether it is really dark or not.

- Have a good look at <https://www.arduino.cc/en/Tutorial/BuiltInExamples/Smoothing> and use this example to create your own outdoor light controller!
- Make sure the threshold level for switching the outdoor light on or off can be set easily!

**Hint:** instead of testing it with an LDR you can test it more easily using a potentiometer.

### 5.2.5 Hysteresis challenge

When controlling an outdoor light like in the previous example, it can happen that during dusk, the outdoor light still switches on and off. The cause is that during dusk the light condition can be very close to the threshold and even the slightest variation (e.g., due to clouds) can make the outdoor light switch.

To prevent this, hysteresis should be used. Implementing hysteresis is a technique often used to prevent redundant switching when a value is varying around a threshold.

- Have a good look at <https://www.aranacorp.com/en/implementation-of-a-measurement-hysteresis-on-arduino/> and make sure you understand the hysteresis concept.
- Apply hysteresis to the outdoor lamp controller of the previous challenge.
- The hysteresis thresholds should be easily adjustable.
- Make sure you test thoroughly!

## 6 Libraries



A library is a collection of functions in a package which can be used for a specific purpose. Often these libraries are freely available and provided by vendors or the online community. Many sensors use a library to communicate with the embedded system. It can be that the library in turn uses a driver for driving the hardware.

See [https://www.youtube.com/watch?v=buFKeqbafDI&ab\\_channel=SimplyExplained](https://www.youtube.com/watch?v=buFKeqbafDI&ab_channel=SimplyExplained) on how to add a library in the PlatformIO environment.

### 6.1 Challenges

#### 6.1.1 DHT Library challenge

- Use the DHT sensor library to read the temperature and humidity with a DHT11 sensor.
- Have a look at the DHT11 datasheet. Can you explain the difference between resolution and accuracy?
- Look closely at the temperature and humidity readings, are they as you expect?
- How fast can you read the temperature?

#### 6.1.2 Ping sensor library challenge

- Use the NewPing library to read the distance with an ultrasonic sensor
- Have a look at the ultrasonic sensor datasheet.
- Look closely at the distance readings, are they as you expect?
- What happens with
  - irregular or sloped objects?
  - moving objects?
  - when no object is nearby?

## 7 Object Oriented Programming (OOP)



At technology we also do OOP. This helps you, among other things, to better maintain your software and make it more readable and also promotes reuse.

OOP is programming language independent, but we think that it is most easy to learn in C#. It is also possible to do OOP on an embedded system like an Arduino. For that you might want to try the expert challenges. However, for the learning outcome, you must program in C#.

### 7.1.1 Introduction Objects en Classes: OOP

- Read and do the [following tutorial](#)
- Read the following references and make the examples to get acquainted with OOP:
  - [Wat is een Class?](#)
  - [Classes gebruiken](#)
  - [Zelf Classes maken](#)
  - [Properties](#)
  - [Je object als string](#)
  - [Voorbeeld pinautomaat](#)
  - [Video over OOP](#)

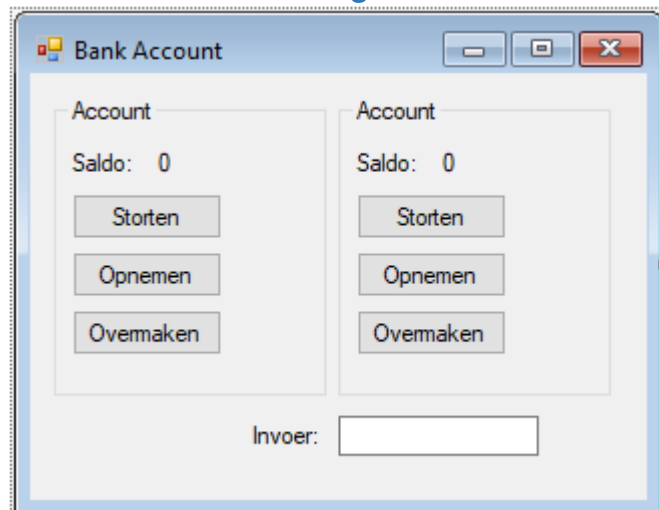
### 7.1.2 Traffic light challenge



You will create a console project in C# with Visual studio .NET that represents a traffic light. Make sure that your program meets the following requirements:

- The project should contain multiple objects of type TrafficLight.
- A TrafficLight can have the colors (states) 'green', 'orange', 'red': use English names for these states.
- For safety, the status is always set to red when creating a TrafficLight.
- From red, the state can only turn to green, then to orange, then to red again.
- You can use a buttonpress to indicate that a car has arrived
- The light will be orange for 2 seconds and green for 5 seconds. Make sure that during the states your system can still react on events

### 7.1.3 Bank account challenge



*Bank account form*

- You will create a Windows Forms application in C# with Visual studio .NET that represents a (highly simplified) bank with two bank accounts. With this application one can:
  - Deposit money into the account on the left or right. Enter the amount in the TextBoxes, then click on the “Deposit” button. Only a positive amount can be deposited. If a negative balance would arise, or if something other than positive numbers are entered, the transaction is not executed and an error message is given by means of a MessageBox.
  - Withdraw money from the account on the left or right. Enter the amount in the TextBoxes, then click on the “Withdraw” button. If a negative balance would arise, or if something other than positive numbers is entered, the transaction is not executed and an error message is given by means of a MessageBox.
  - Transfer money from left to right (or right to left). In the TextBoxes on the left (right) you enter the amount to be transferred and then click on the Button >> (<<). If a negative balance would arise with the paying party, or if something other than positive numbers has been entered, the transaction will not be executed and an error message will be given by means of a MessageBox. The displayed balance is of course neatly adjusted after every transaction.

Show in this application the concepts that you learned about OOP.

A few mandatory requirements are:

- UI code is separated from the other code
- No public fields
- Two objects must be made from one Bankaccount class
- The Bankaccount class implements the methods deposit, withdraw and Transfer to
- The bank account implements a constructor with several parameters

- Each account gets a unique number, you could make a AccountNumberGenerator class for that.

**Tip:** use the bank account form image as an example

#### 7.1.4 LED class expert challenge



Doing OOP on an Arduino is a little bit different than doing OOP in C#, so please take a look at the *Workshop OO on Arduino* in the Toolbox section of this course and the following internet resource <https://docs.arduino.cc/learn/contributions/arduino-creating-library-guide>.

**Tip:** There is an 'Arduino\_OOP\_Demo\_Basic' example program in the Toolbox - > Example programs section of this course which contains some do and don'ts when programming OOP on an Arduino!

- If you are into OO programming on the Arduino, create a LED class which can blink a LED. With this class you can handle as many LEDs as you like just by creating instances of the LED class.

#### 7.1.5 Button class expert challenge

- If you are into OO programming on the Arduino create a button class so you can handle as many buttons as you like.

#### 7.1.6 Timer class expert challenge

**00:00:00**

- If you are into OO programming on the Arduino, have another look at the Blinking LEDs challenge above.
- Make your own timer class. Then use this timer class to improve the Blinking LEDs challenge. Note that using a timer class will make your code more readable, reusable and scalable!

#### 7.1.7 Traffic light expert challenge



- Make a traffic light controller on the Arduino and use classes

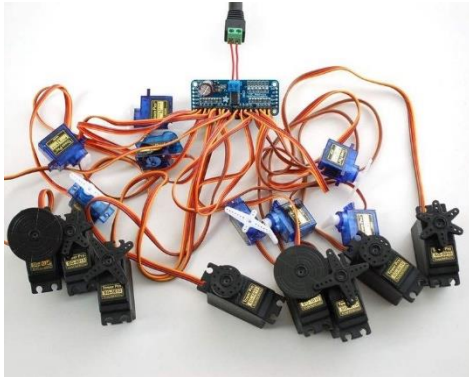
#### 7.1.8 Car system expert challenge

If you are into OO programming on the Arduino simulate a car with two subsystems: (1) the steering wheel has three LED lights, two buttons, temperature sensor, light sensor and a potentiometer, and (2) the touch screen has two text fields and a list box. The steering wheel embeds an Arduino and a Rich shield, whereas the touch screen is a laptop with a Windows app. Both subsystems communicate with each other using a serial connection.

- The potentiometer simulates the steering wheel. You can use the full scale of the potentiometer.
- The two buttons are used to indicate in which direction the user want to turns. The left button lets the yellow LED blink until the user steers from the left to the centre (or a small percentage in the right half). The right button lets the blue LED blink until the user steers from the right to the centre (or a small percentage in the left half). This behaves like a real car. Be aware that the yellow and blue LEDs never blink at the same time.



- The headlights are automatic and they turn on when it's dark, otherwise they're off. The green LED simulates the headlights. The headlights may never flash due to a cloud  
The touch screen is a C#.net forms application. A text fields shows the temperature and headlight status. The list box is used to collect activities and acts as a black box of the car. You must determine what information is important to be collected.



## 8 Actuators

### 8.1 Pulse width modulation (PWM)

For an explanation on PWM, see

<https://www.halvorsen.blog/documents/technology/resources/resources/Arduino/Arduino%20PWM.pdf>.

### 8.2 Challenges

#### 8.2.1 PWM LED challenge

- Connect a potentiometer and a LED to your Arduino.
- Write a program such that you can control the brightness of the LED with the potentiometer.
- Show your loop skills to create a light show using an RGB LED. Use the potentiometer and buttons to change the behavior of the light show, e.g. the pattern, speed, brightness.

**Hint:** you can use the `map()` function to determine the influence of the potentiometer.

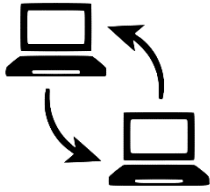
#### 8.2.2 Servo control challenge



In this challenge you are going to control a servo with great precision. This is needed for example in robot assisted surgery.

- Connect a servo and two sensors like a potentiometer. One sensor is for coarse control of the servo (to get it in roughly the right position) and the other one for fine (high precision) control to do the surgical movements.
- Write a program to control the servo with both sensors.
- Can you improve your program by make the servo control easier for a surgeon with shaking hands?

## 9 Communication



For many applications in the Technology profession, communication between equipment is essential. Computer networks, Smartphones, remote controls and smart energy meters are examples of this.

Just like people talking to each other, communication between devices requires agreements. Without these agreements, communication is not possible. For example, you must speak and be able to understand the same language. And it's helpful if you know who can talk and when.

To find out in which ways you can communicate with the Arduino take a look at the following internet resource <https://www.deviceplus.com/arduino/arduino-communication-protocols-tutorial/>.

### 9.1 Challenges

#### 9.1.1 Remote control challenge



In the orientation phase you got acquainted with receiving a (serial) message from a PC.

- If needed you can take a look at the following link again.  
<https://www.arduino.cc/reference/en/language/functions/communication/serial/read/>
- Take a look at the workshop *Serial communication with the Arduino* in the Toolbox section of this course
- Now you will remote control a (red) LED connected to the Arduino from the PC using messages that meet an agreed format. When the Arduino receives the message "#SET\_LED\_ON%" from the serial port, the LED will be turned on. Use the String type on the Arduino to save and compare a received message. For String usage, see <http://arduino.cc/en/Reference/StringObject>.
- What are the '#' and '%' characters for?
- Extend your program with two other LEDs and modify your code so that you have a working 'traffic light'.
- Make sure that your 'traffic light' can be controlled from your PC. Use a command with a parameter like "#SET\_LED:<on\_off, led\_color>%"
- Make sure that your program handles unknown values for led\_color and that the sender of the message is informed that the value is unknown

Hints:

- Use the Serial Monitor in PlatformIO to send messages to the Arduino. With the Serial Monitor you can always test the serial communication.
- You can use `String.startsWith(...)` to determine the type of message.
- You can use `String.substring()` to 'cut out' the number (the parameter).

- A text to int conversion function is provided.

### 9.1.2 controlled testing expert challenge

- Modify the self-test challenge so, that it can be controlled via a C# program via serial communication.

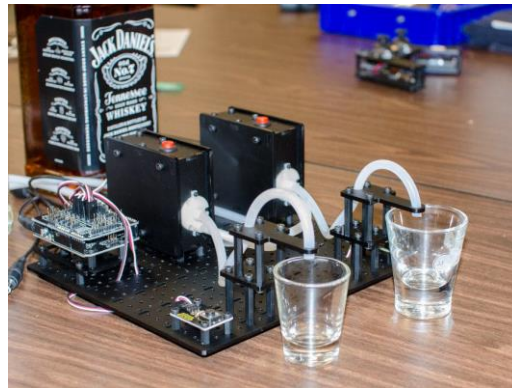
### 9.1.3 Arduino 2 Arduino serial communication expert challenge

- With a fellow student, connect two Arduino's to each other using only RX, TX and ground. Can you send messages back and forth?
- What is the maximum speed?
- Define your own protocol for the communication. See *Serial communication with the Arduino* in the Toolbox section of this course

### 9.1.4 Arduino 2 Arduino I2C communication expert challenge

- With a fellow student, connect two Arduino's to each other using only SDA, SCL and ground. Can you send messages back and forth?
- What is the maximum speed?
- Define your own protocol for the communication. See *Serial communication with the Arduino* in the Toolbox section of this course.

## 10 Project



During the 'verdieping' you will also work on the 'Talk to me' project. There you can demonstrate the different parts of the learning outcome in a bigger context. It should not be difficult to add enough Technology to the project so that all parts of the learning outcome are covered.

To summarize:

### **Interactive embedded systems**

- Make your embedded system fully integrated and interactive by realizing a reliable (two-way) communication between your embedded system and the host.

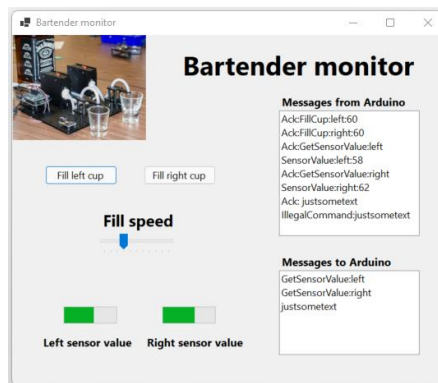
### **Programming**

- Show your imperative and object-oriented programming skills!
- Typically, you can do that in C on the embedded board and in C# on the host computer.
- You can also use Object Oriented C++ on the embedded board. Only do this when you are confident you can handle this, but make sure you also know how to do things in C#.
- In C# you could for example create a monitor application to fully test the embedded system.

With this monitor application you could for example:

- Receive messages coming from the embedded system.
- Send messages to the embedded system.
- Read sensor values.
- Set (or override) actuator values.

- Thoroughly test your two-way communication protocol, also with messages not following the protocol!



Hint: put the communication part in a separate class so your fellow students can simply use your code in their host application.

An additional advantage of this approach is that you do not have to wait for your host application colleagues or vice versa.

### Sensors and actuators

- Impress the user of the project by using sensors and actuators to create a great user experience!

### Different I/O techniques

- When using different sensors and actuators, you will automatically come across different I/O techniques.

*Good Luck!*