



# Electronics Project

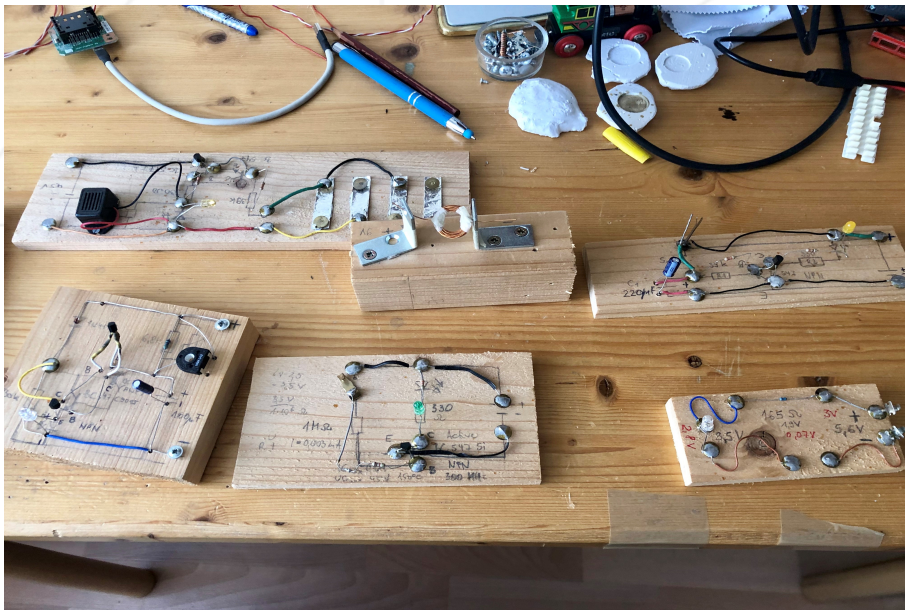
## Day00: First Programs

[contact@42chips.fr](mailto:contact@42chips.fr)

*Summary: Once upon a time, there was a little spectrum...*

# Chapter I

## Prologue



At the beginning of radio, amateurs would nail bare copper wire or terminals to a board and solder electronic components to them.

Sometimes a schematic diagram on paper was first glued to the board as a guide to place the terminals, then components and wires were installed at the associated symbols on the diagram.

The use of tacks or small nails as mounting posts was also common.

Breadboards have evolved over time, the term now being used for all kinds of electronic device prototypes.

The most commonly used breadboard today is usually white plastic and allows components to be plugged in without the need for solder.

It was designed by Ronald J. Portugal in 1971 and is now commonly used by everyone to prototype project parts.

# Chapter II


## General instructions

Unless explicitly stated otherwise, the following instructions will be valid for all assignments.

- The language used for this project is C.
- It is not necessary to code according to the 42 norm.
- The exercises are ordered very precisely from the simplest to the most complex. Under no circumstances will we consider or evaluate a complex exercise if a simpler one is not perfectly successful.
- You must not leave any files other than those explicitly specified by the exercise instructions in your directory during peer evaluation.
- All technical answers to your questions can be found in the **datasheets** or on the Internet. It is up to you to use and abuse these resources to understand how to complete your exercise.
- You must use the datasheet of the microcontroller provided to you and comment on the important parts of your program by indicating where you found the clues in the document, and if necessary, explaining your approach. Don't write long blocks of text, keep it clear.
- Do you have a question? Ask your neighbor to the right or left. You can ask in the dedicated channel on the Piscine's Discord, or as a last resort, ask a staff member.

# Chapter III


## Setup

	Exercise 00
Makefile	
Turn-in directory : <i>ex00/</i>	
Files to turn in : <b>Makefile</b> , <b>main.c</b>	
Allowed functions : <b>avr-gcc</b> , <b>avr-objcopy</b> , <b>avrdude</b>	
Notes : n/a	

- The **main.c** file should contain a **main()** program that does not contain any instructions.
- Write a **Makefile** file with the **all** rule that executes the **hex** and then **flash** rules.
- The **hex** rule compiles the **main.c** file into **main.bin** with an **F\_CPU** variable to select the microcontroller frequency. Then it generates the **main.hex** file from the **main.bin** file.
- The **flash** rule copies the **main.hex** file into the microcontroller's flash memory.
- The **Makefile** must also implement the **clean** rule that deletes the **main.hex** and **main.bin** files.

# Chapter IV


## Let there be light

	Exercise 01
A glimmer of hope	
Turn-in directory : <i>ex01/</i>	
Files to turn in : <code>Makefile</code> , <code>main.c</code>	
Allowed functions : <code>avr/io.h</code>	
Notes : n/a	

- You must write a program that turns on the LED D1 (PB0).
- You must use only AVR registers (DDRX, PORTX, PINX).



You must explain the function and values assigned to the registers in comments every time!


	Exercise 02
Blinker	
Turn-in directory : <i>ex02/</i>	
Files to turn in : <b>Makefile</b> , <b>main.c</b>	
Allowed functions : <b>avr/io.h</b>	
Notes : n/a	

- You must write a program that turns on and off the LED D1 (PB0) at a frequency of 1Hz.
- You must write code that allows you to wait for several hundred milliseconds and which will be inserted into the infinite loop of the program (no hardware TIMER).
- The change in state of the LED must be made with a single [bitwise operation](#), do not use a condition (if else).
- You must use only the AVR registers (DDRX, PORTX, PINX).




1Hz == (on for 0.5 sec and off for 0.5 sec)

The exercise will be considered invalid if your main returns.

	Exercise 03
Lumos	
Turn-in directory : <i>ex03/</i>	
Files to turn in : <b>Makefile</b> , <b>main.c</b>	
Allowed functions : <b>avr/io.h</b> , <b>util/delay.h</b>	
Notes : n/a	

- You must write a program that turns on LED D1 (PB0) when button SW1 (PD2) is pressed.
- When the button is released, the LED turns off.
- You must use only AVR registers (DDRX, PORTX, PINX).

	Exercise 04
Day, Night, Day, Night	
Turn-in directory : <code>ex04/</code>	
Files to turn in : <code>Makefile</code> , <code>main.c</code>	
Allowed functions : <code>avr/io.h</code> , <code>util/delay.h</code>	
Notes : n/a	

- You must write a program that reverses the state of the PB0 LED each time the SW1 button (PD2) changes from the `released` state to the `pressed` state.
- You must use only AVR registers ( `DDRX`, `PORTX`, `PINX` ).




Be careful of `bounce effects`!



# Chapter V

## It's not rocket science !

	Exercise 05
Binary Counter	
Turn-in directory : <i>ex05/</i>	
Files to turn in : <b>Makefile</b> , <b>main.c</b>	
Allowed functions : <b>avr/io.h</b> , <b>util/delay.h</b>	
Notes : n/a	


- You must write a program that:
  - increments a value each time you press button SW1
  - decrements a value each time you press button SW2
  - displays this value in binary on LEDs D1 D2 D3 and D4 permanently.
- You must use only the AVR registers ( **DDRX**, **PORTX**, **PINX** )



If the 4 LEDs were on GPIO PB0, PB1, PB2, PB3, this exercise would be simpler.

Unfortunately, LED D4 is on PB4 instead of PB3 because PB3 is used for something else.

You will have to manipulate bits.

	Exercise 06
Knight Rider	
Turn-in directory : <i>ex06/</i>	
Files to turn in : <b>Makefile, main.c</b>	
Allowed functions : <b>avr/io.h, util/delay.h</b>	
Notes : n/a	

- You have to write a program which redoes with LEDs the K.I.T.T. animation of Knight Rider
- The animation consists in lighting a single LED and making it move from right to left
- You must use only the AVR registers (DDRX, PORTX, PINX)



Knight Rider but what is this thing?