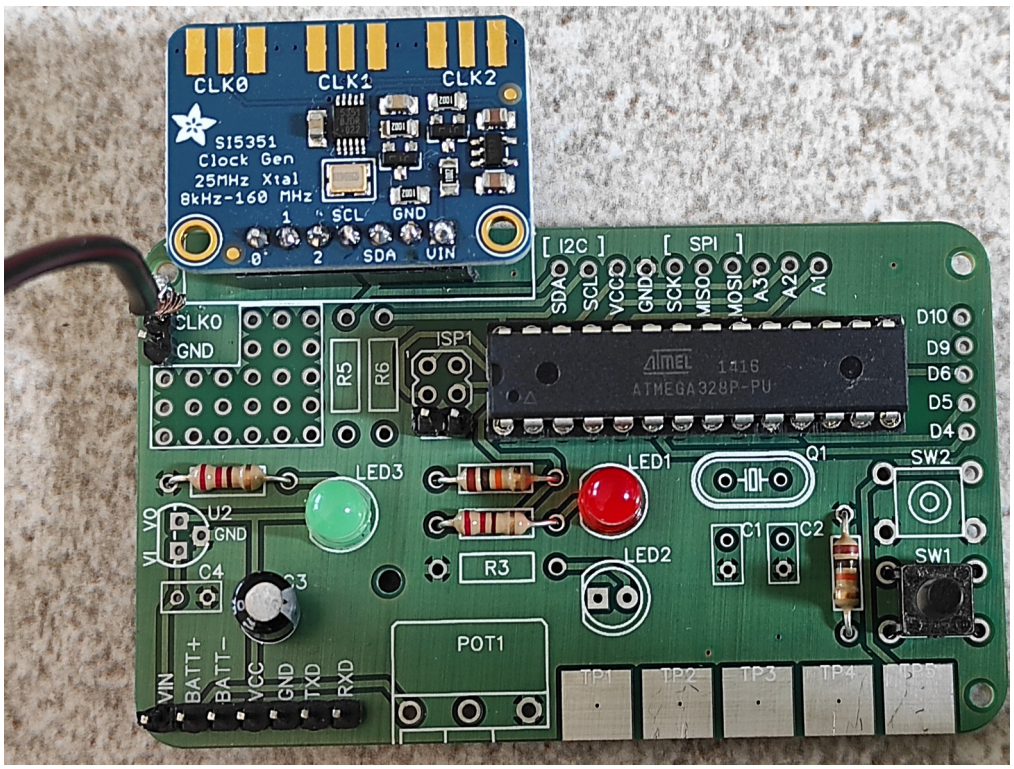
LOW POWER ARDF TX v1.0 “WABBIT”

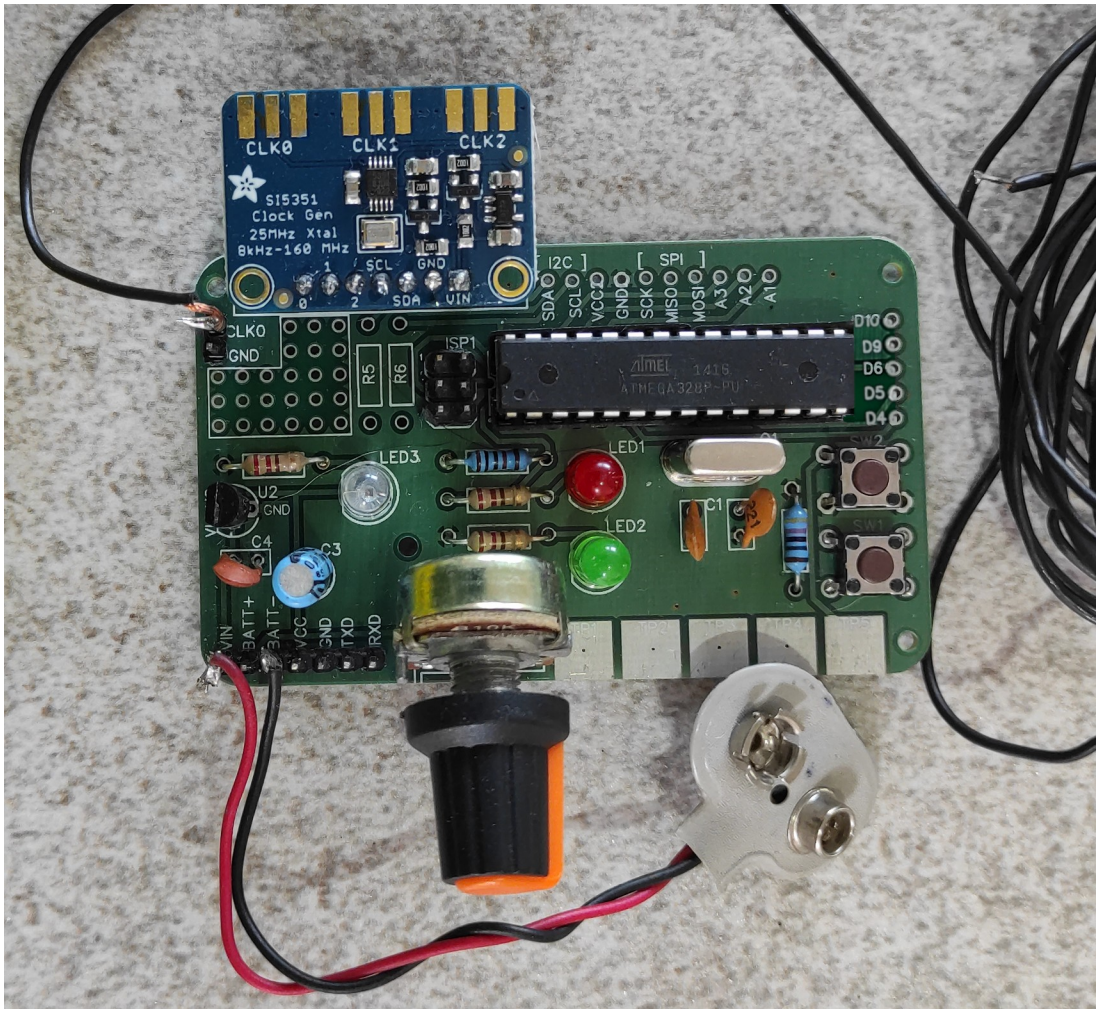
Low power ARDF TX “Wabbit” is a learning tool for electronics, microcontrollers and hamradio. It is an Arduino based board which uses an Atmega328p microcontroller paired with Si5351 DDS module to generate any frequency between 8kHz and 150MHz.
Finished Wabbit can be used as a beacon for ARDF or any other type of activity which could use transmitter with manual or automatic control – eg. a beacon to test propagations, tunning antenna transmitter or for learning Morse, either in transmitting or receiving.

This is a small Wabbit:



and in the next few days you will build one!

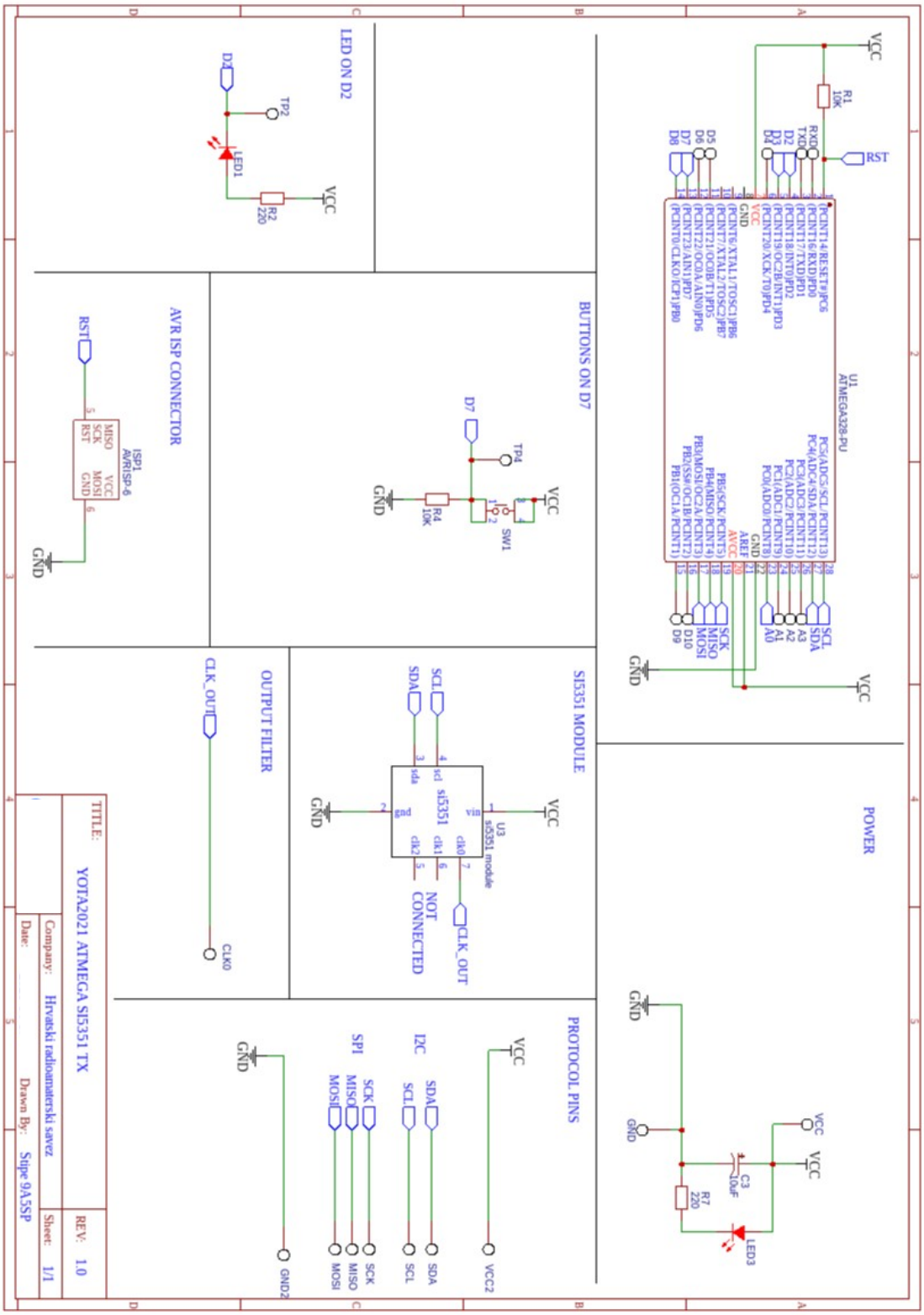
If you look at the picture of the board you can see that some parts are missing, their places are not populated. The board above is the minimum needed for making an automated transmitter. The full board would look something like this big Wabbit:



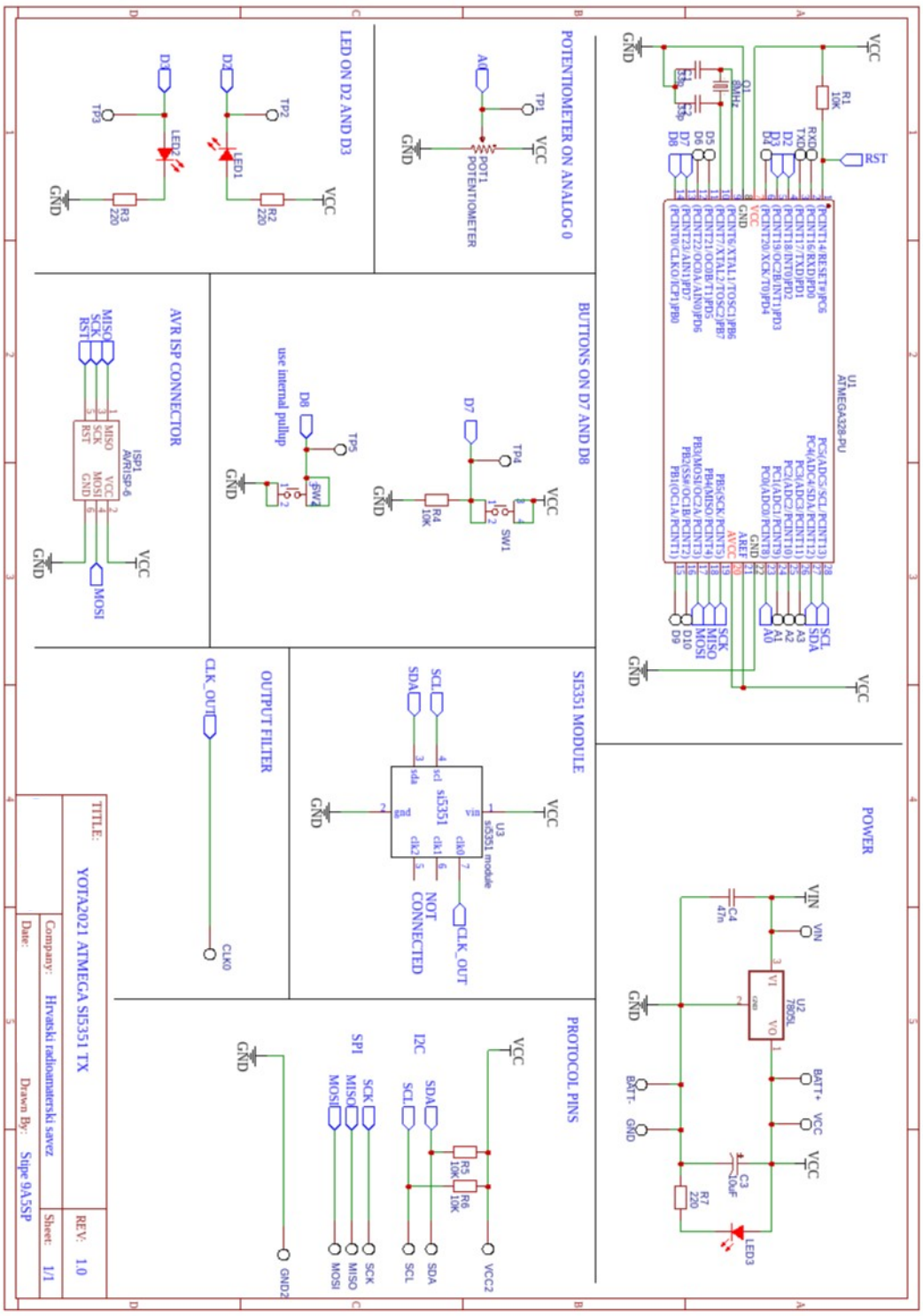
Each Wabbit is different, by it size (number of elements), looks (how the elements are oriented and soldered) and brains (the code in the microcontroller).

Compare the schematics of a minimum board (small Wabbit) and the full board (big Wabbit), and see what we can live without.

Small Wabbit



Big Wabbit



Big Wabbit has a potentiometer, second LED, second switch, a voltage regulator, pull up resistors on I2C lines and an AVR ISP connector!

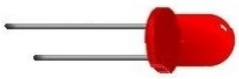
All of those extra elements is not needed to have a Wabbit.
It’s smaller, but it can still kick a mean punch!

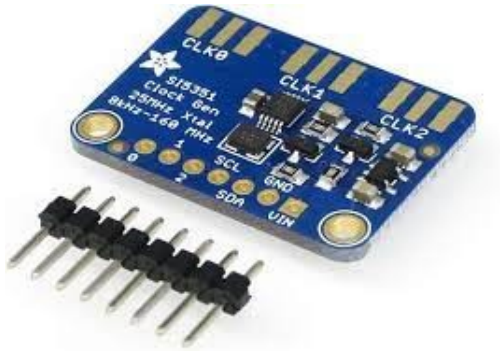
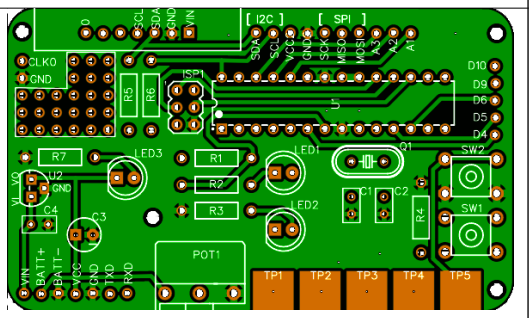
So let’s build you a small Wabbit. You should have received a bag with a bunch of elements, something like this:

YOUNGSTERS ON THE AIR

9A22YOTA – Croatia Low Power ARDF TX v1.0 "Wabbit"

You should have these elements to build a small Wabbit:

Name	Value	Pieces	Picture
Electrolit capacitor C3	10 uF	1	
Integrated circuit socket U1	28 pins	1	
Resistors R1 and R4	10 kΩ	2	 Brown,black,orange
Resistors R2 and R7	220 Ω	2	 Red, red, brown
LED (LED1)	red	1	
LED (LED3)	green	1	
Pushbutton SW1		1	
Header „male"	2 pins	2	
Header „male"	7 pins	1	
Header „female"	7 pins	1	

Si5351 module		**1**	
Wabbit PCB		**1**	

What's missing in the bag:

Integrated circuit IC1	**ATMega328p**	**1**	
Wire antenna		**1**	

ATMEGA328p is very sensitive to static electricity. You need to be very careful when it's not connected in the Wabbit, so we will keep it safe until the programming workshop.

I dont' have the same elements as others do?
In the bag with your Wabbit you will find additional components, needed to build a big Wabbit. Don't be alarmed if you have more or less than your colleagues – if you miss some of the components just ask your instructor for a replacement.

Why is there an 8MHz crystal if it's not used in a small Wabbit?
The microcontroller is internally controlled by an 8MHz oscillator. This is OK for a normal use, but functions involving time measuring(eg. delay()) will not be absolutely correct (eg. for 1000ms delay it will be 997 or 1003).
If time accuracy is important then the crystal and two capacitors should be placed at Q1, C1 and C2, and fuses on ATMEGA328 should be changed.

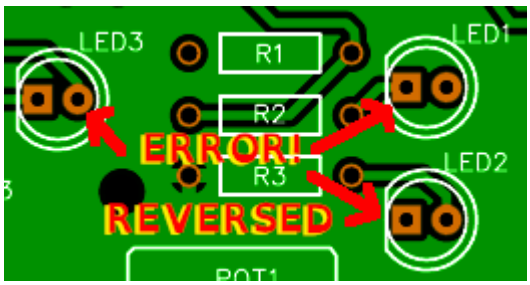
Why is the used frequency (even with the crystal) only 8MHz?
The ATMEGA328 works reliably on 8MHz all the way down to 3V, while the Arduino 16MHz boards work only on 5V. When the whole system works on 8MHz, the Vcc voltage can be anything between 3 and 5V without a problem, which is ideal for different power sources – power bank on 5V, 3 AA batteries which give 3.6(NiMH) to 4.5V(alkaline) or 3.7V LiIon battery.

SOLDERING

If you're experienced in soldering:

You can continue on your own soldering the small Wabbit (minimal board):

There is an error in silkmask for LED's (LED1, LED2 and LED3) – the silkmask is reversed!



If you're unexperienced in soldering:

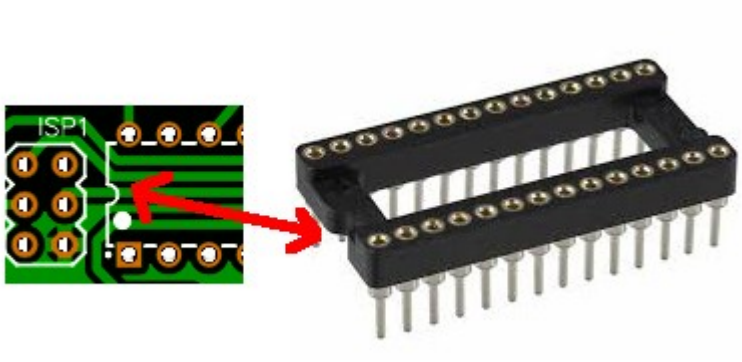
Soldering is a basic skill which is easily learned, and is present in everyday hamradio life. Still, be careful, take every precaution, as you'll work with a very hot object(s)!

Remember: If it smells like chicken, you're holding it wrong!

A short comic: **Soldering is easy**, is available on the Internet. You can Google it, or jump straight to it:

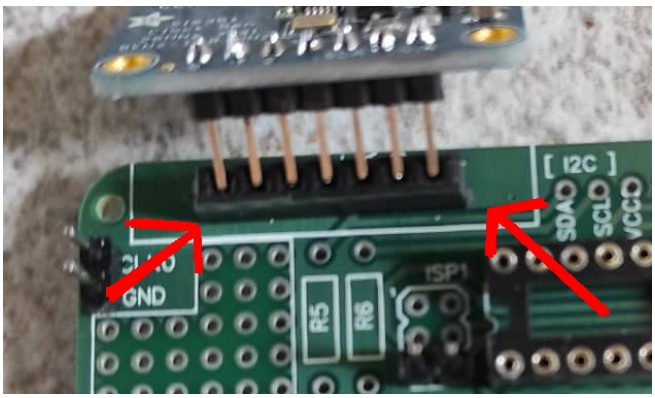
https://tinyurl.com/soldeasy

Step	Description
1.	Begin by soldering the Integrated circuit socket U1 (28 pins). Be careful to put the notch on the socket (see picture) in the same orientation as drawn on the silkmask of the board. Tip: Solder the last elements diagonally (bottom left, top right) before continuing to other points. This way the socket will not move while soldering.

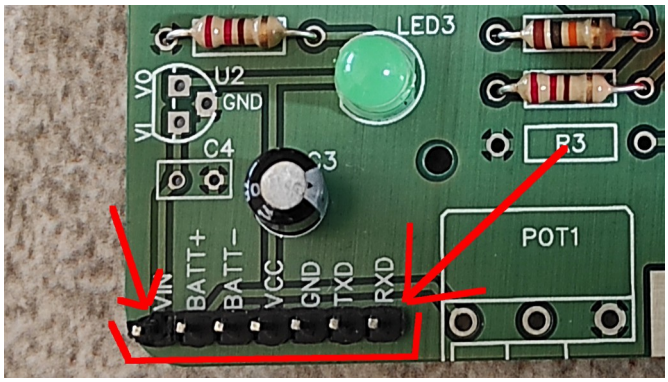


CONGRATULATIONS, YOU'VE SOLDERED YOUR FIRST PART OF WABBIT!

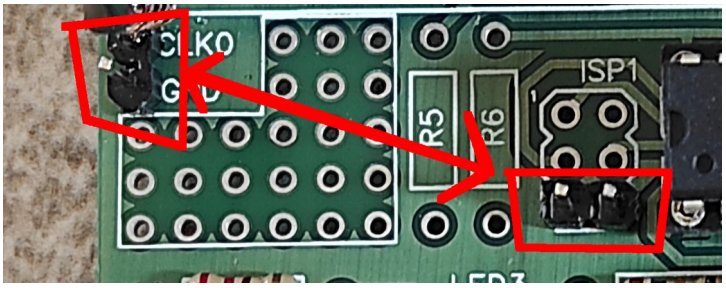
2. Solder the female 7-pin header in the top left header place.
   Tip: solder the ending elements first, then solder the rest.



3. Solder the male 7-pin header in the bottom left header place.
   Tip: solder the ending elements first, then solder the rest.



4. Solder the two 2-pin male headers for CLK0 / GND, and on the bottom of ISP1.



5. Open the bag for Si5351 and solder the male header to the Si5351 module. **Check for orientation so there is matching of pins and header!**

6. Solder the resistors R1 and R4. They are 10 kΩ ones:

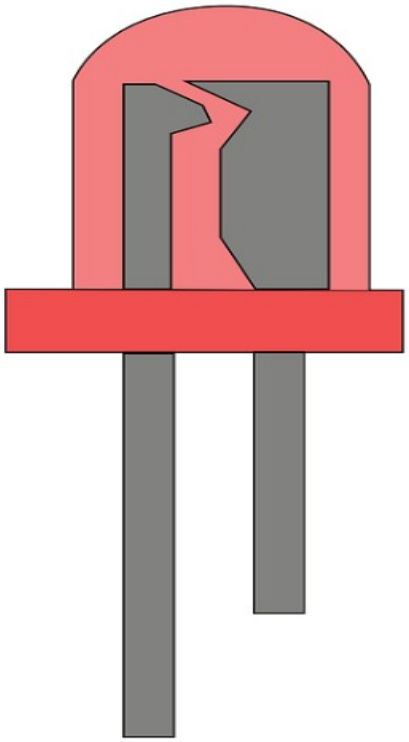
Brown,black,orange

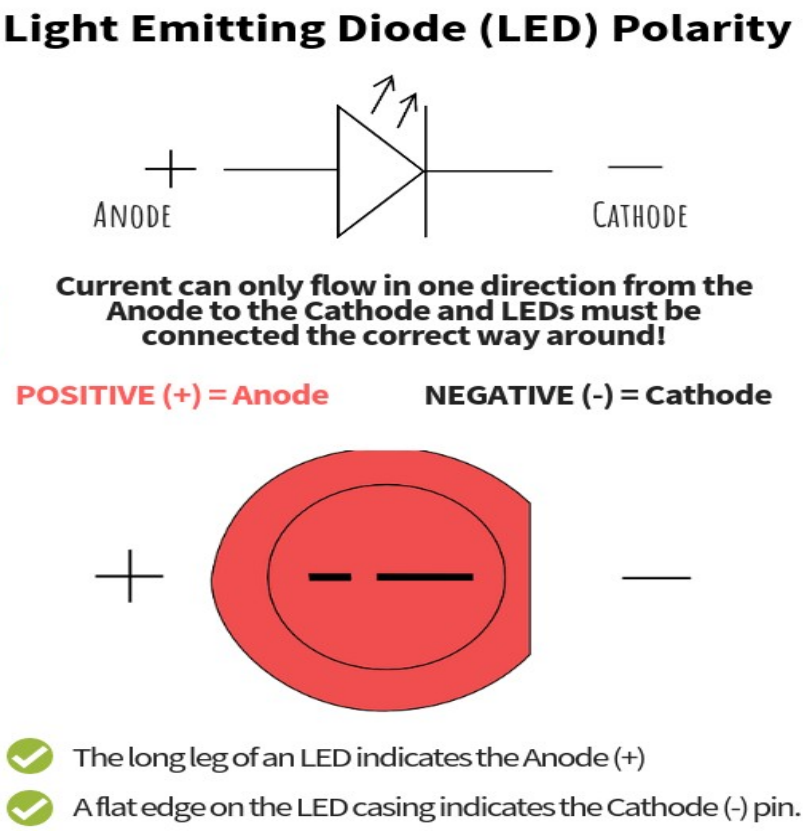
7. Solder the resistors R2 and R7. They are the 220 Ω

Red, red, brown

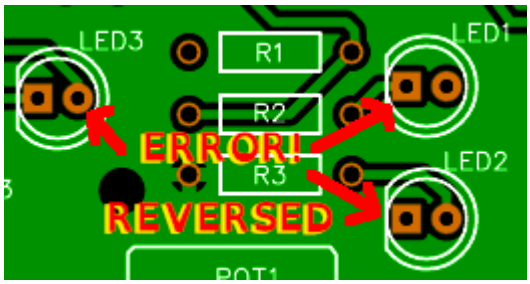
8. Solder the switch SW1 (the bottom one).
Note that it should only go one way (or upside down), as rotated it's not wide enough to get into the holes.

9. Solder the LED1 and LED3.
Note that they have legs of different length, and that one side is flat while the other is rounded.





There is an error in silkmask for LED's (LED1, LED2 and LED3) – the silkmask is reversed!



10. Solder the electrolytic capacitor C3.
    Electrolytic capacitor is polarized capacitor, it has a positive and negative side.
    Note that it has a '-' on silkscreen, and there is a blue-gray strip on the capacitor with '-' marks.

11. Solder the wire of the antenna (wire provided by the Instructor) to the 2-pin header at CLK0.

CONGRATULATIONS, YOU'VE SOLDERED A SMALL WABBIT!

I'm finished, but there is still a lot of time?
You can solder potentiometer from the bag to POT1, and ask instructor for second LED (LED2) and 220 Ω resistor (R3), second switch (SW2), or a voltage regulator (U2).

Should I solder the crystal quartz Q1 and capacitors C1 and C2?
You can, but it's useless unless the ATMEGA328 has correct fuses to use the crystal. The crystal is needed only if correct timing is necessary.

Should I solder the resistors R5 and R6? They are not in the bag?
Resistors R5 and R6 are used as pull up resistors for I2C SDA and SCL lines. The Si5351 module has these resistors on those lines on the module itself, so it's not necessary to solder them to the Wabbit.
Solder R5 and R6 if some other board is connected to I2C lines!

How can I test my Wabbit?
You can power your Wabbit on the VCC pin with voltage 3-5V, and test power led LED3 (don't forget the GND pin also).
If you don't have your ATMEGA328 in the socket, you can test other elements by short connecting testing points 1-5 (TP1-TP5) to Vcc, GND or between themselves – ask instructor how to do that.

What's next?
Next will be giving your Wabbit a brain in the programming workshop – you will make the code for the microcontroller to transmit your callsign to the world!

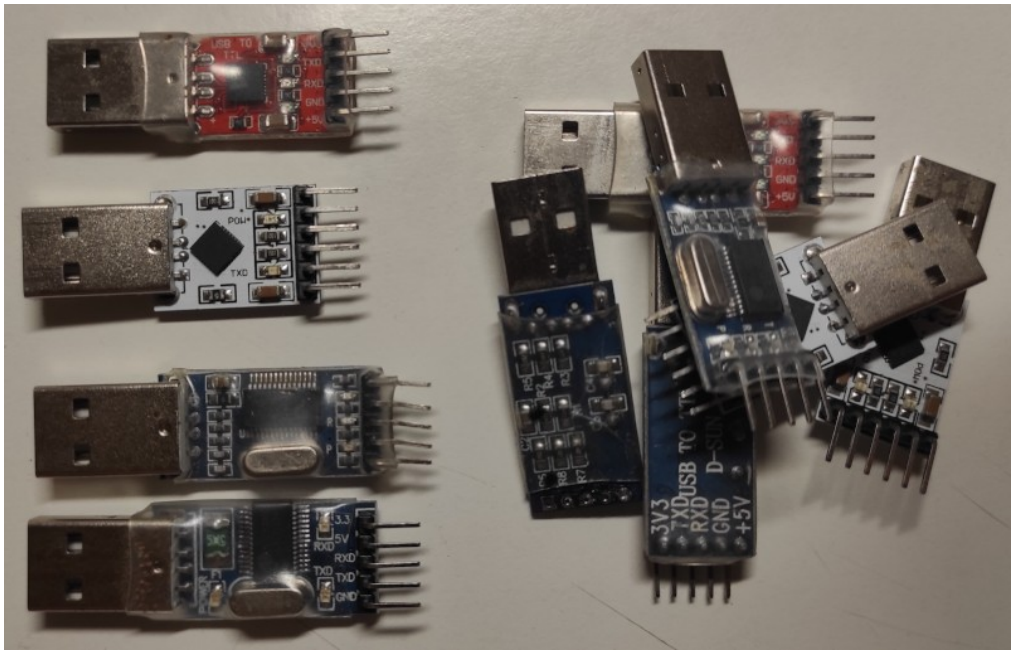
PROGRAMMING

It's time to give some smarts to your Wabbit. It can do a lot, it just depends on the size of your Wabbit (did you make your Wabbit big or small?), and your imagination.

For starters, you need to get a brain for your Wabbit. It's an Atmel ATMEGA328p microcontroller which has 28 pins. The ATMEGA328 has a notch on one side and indented dot, and it should be turned to the left (if you soldered the socket correctly, it should match the socket).

ATMEGA328 is a microcontroller, which means a whole microcomputer is put into one chip – it has 32 kB of memory for your programming code (ROM), 2 kB of active memory for your variables (RAM), it has 23 input output lines (to talk with the world) out of which 6 can be analog 10-bit inputs (can detect any voltage between 0 and Vcc volts (eg. 5V) in 1024 small increments), and it has I2C and SPI support, it has a way to measure time when it's on etc ...
The programming possibilites of ATMEGA328 are endless!

After getting the brains, Wabbit needs to be connected to the PC. To do it, a USB to TTL converter (or adapter) is used. Most popular are the converters (adapters) from the company FTDI, but in the cheap far eastern companies produce their own version which are used in a lot of devices- Silicon Labs CP210x, Prolific 2303 and CH340.

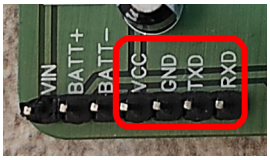


You may need to install the drivers for the device you're using. Be careful on source of drivers!

The connections on the Wabbit are on the 7 pin header:

- **VCC** – voltage of 3.3 or 5V on the USB-TTL converter
- **GND** – ground
- **TXD** – connect it to RX (or RXD) on USB-TTL converter*
- **RXD** – connect it to TX (or TXD) on USB-TTL converter*

* on some converters it's in reverse, so if it's not working reverse these two wires



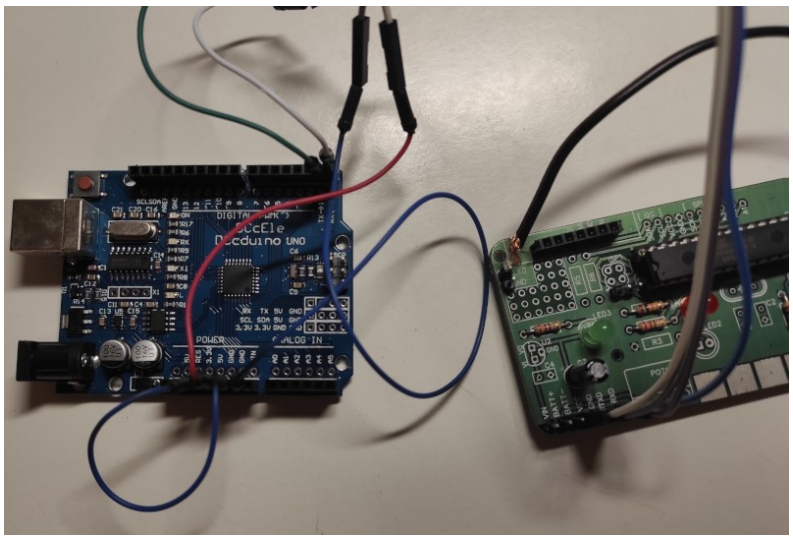
By connecting the USB converter and the Wabbit to your computer you can make serious damage if either of two devices are malfunctioning, due to incorrectly or badly soldering, or any other reason.

IN CASE OF ANY DOUBT IN YOUR WABBIT OR USB CONVERTER DO NOT CONNECT IT TO YOUR COMPUTER. USE BRAIN TRANSPLANT INSTEAD!

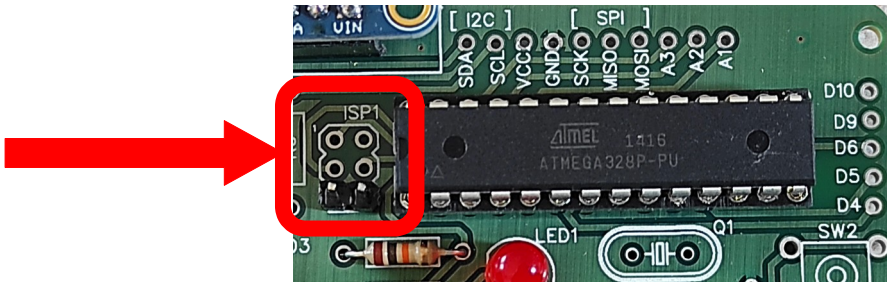
Brain transplant is a procedure where code is programmed into ATMEGA328 on a healthy Wabbit, then the microcontroller is removed from that wabbit and put into a sick one.

Full Arduino ATMEGA328 based boards have the USB-TTL converter on themselves, and they could be used to program the Wabbit:

- Connect all the wires the same as with USB-TTL converter
- add a wire between RESET and GND to stop the ATMEGA328 on the Arduino board to get in the way while programming the Wabbit.
-



Alternatively you can use a full Arduino board as an ISP programmer and connect it to ICSP pins on the Wabbit (6 pin header left to the microcontroller chip, you soldered a 2 pin header there). That way you could change the fuses of the microcontroller to use the crystal which would give your Wabbit accurate timing. But this practice is out of scope of the workshop, and is left for you to explore.



Before connecting the USB-TTL converter to the computer please check if you have installed:

- Arduino software, version 1.8.0 or later
- drivers for the appropriate USB-TTL converter

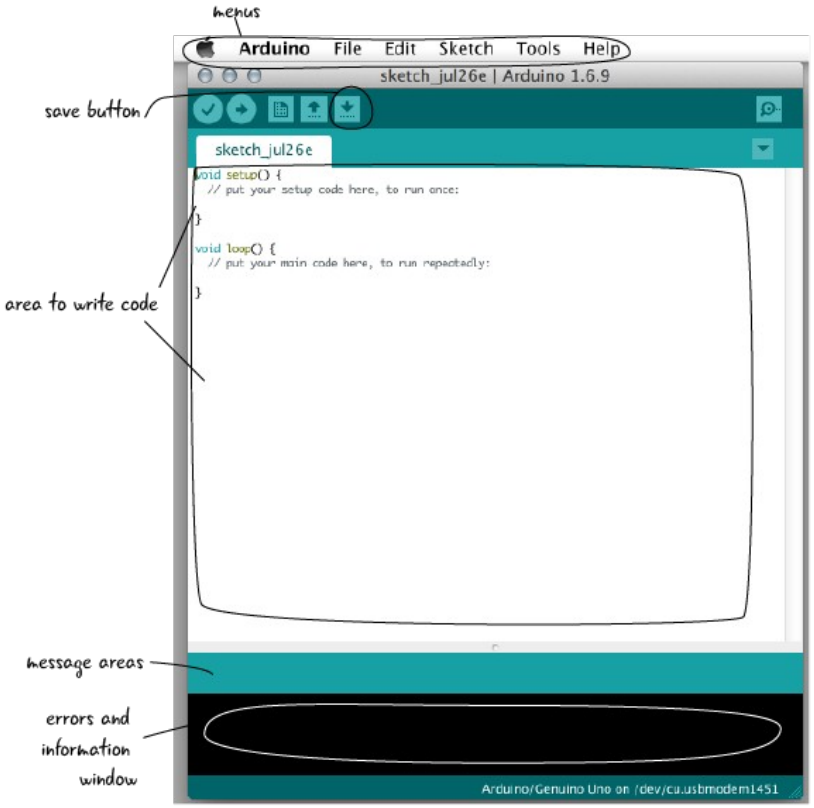
If unsure, call instructor.

Unplug the Si5351 oscillator module at the beginning of the workshop . We will use it in the second part of the workshop.

LED3 (the power LED) should turn on when the USB-TTL converter is connected to the Wabbit and to the computer - if the Wabbit is healthy.

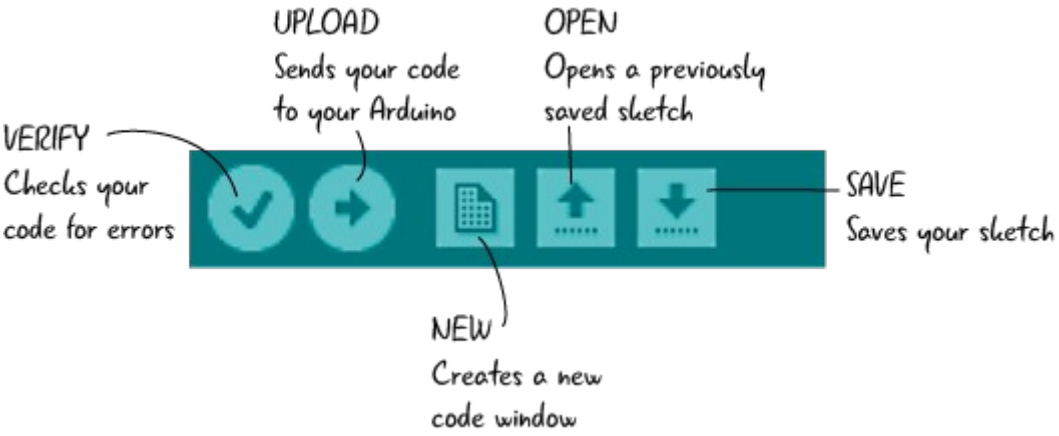
If LED3 doesn't turn on immediately after plugging the USB-TTL converter into the computer – UNPLUG USB-TTL converter from the computer and ask for help!

Now start Arduino program on the computer. You should get a window like this:



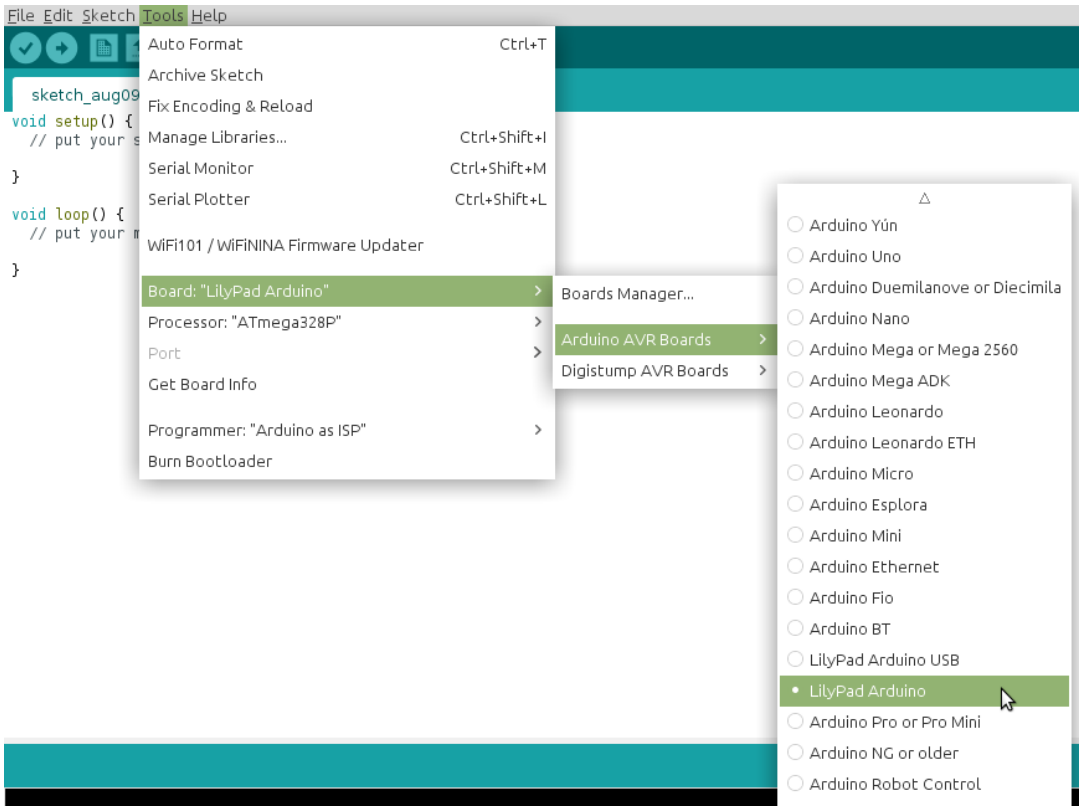
Arduino code is done in something called sketches – one sketch is one program (regardless of how many basic or additional files are needed).

The top line in the Arduino window is import as it gives us a lot of options:

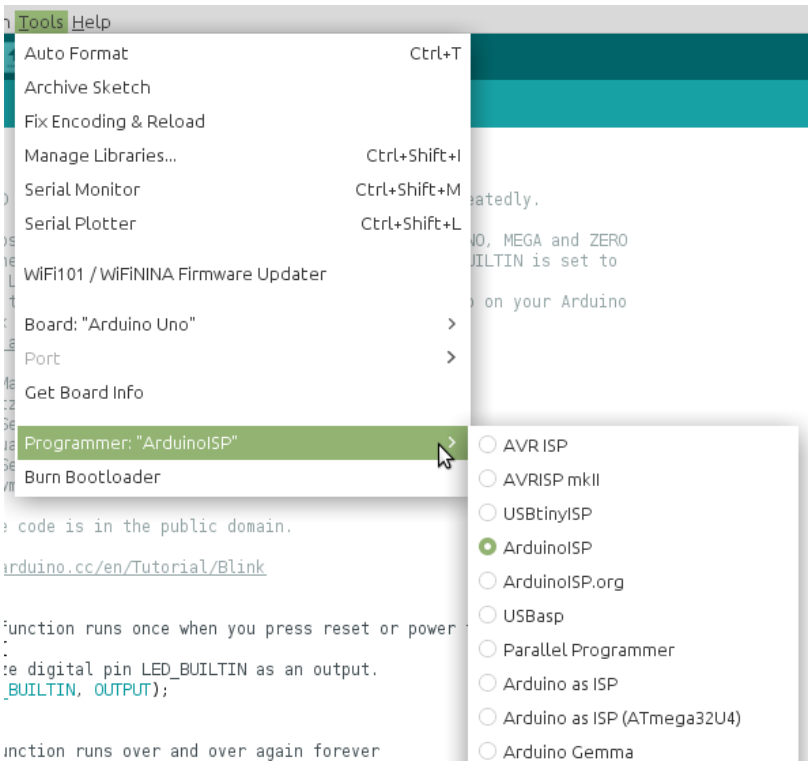


Before we start programming we need to do one more thing: choose the board. Wabbit is based on the Arduino boards which don't have the external oscillator, but rely on internal 8MHz oscillator.

So go to Tools → Board → **LilyPad Arduino** (or in some cases Tools→ Board→ Arduino AVR Boards→ **LilyPad Arduino**), and select a port (right under the Board).



You need to choose Programmer: select value Arduino ISP



Great, now were heading somewhere.

Let’s try the simplest program, BLINK!

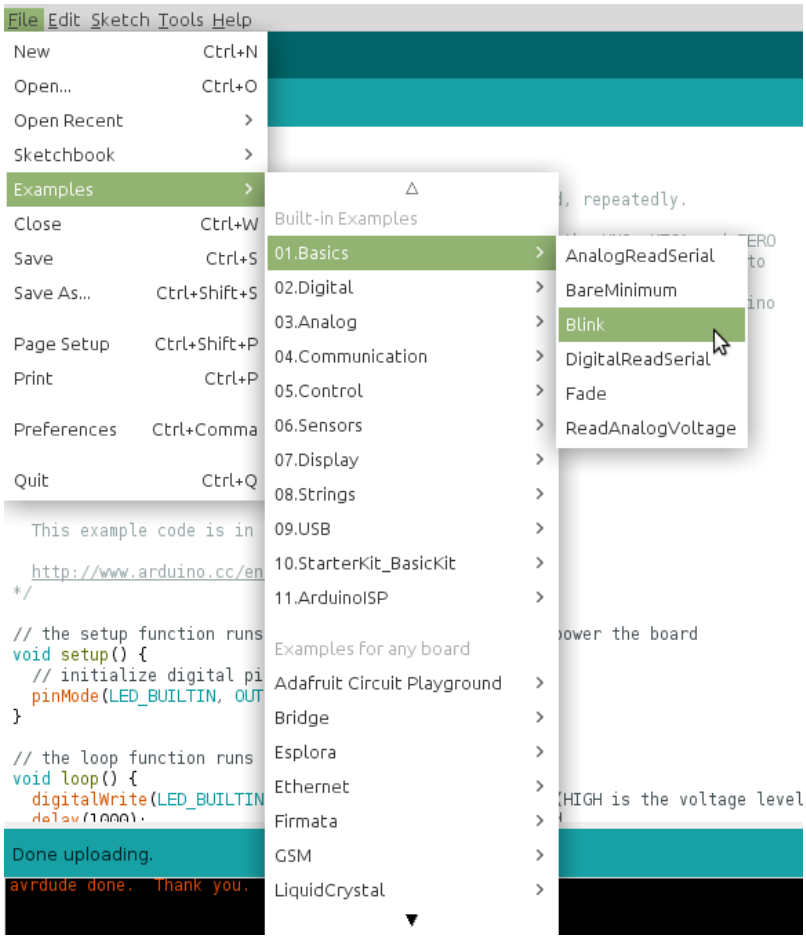
YOUNGSTERS ON THE AIR

9A22YOTA – Croatia

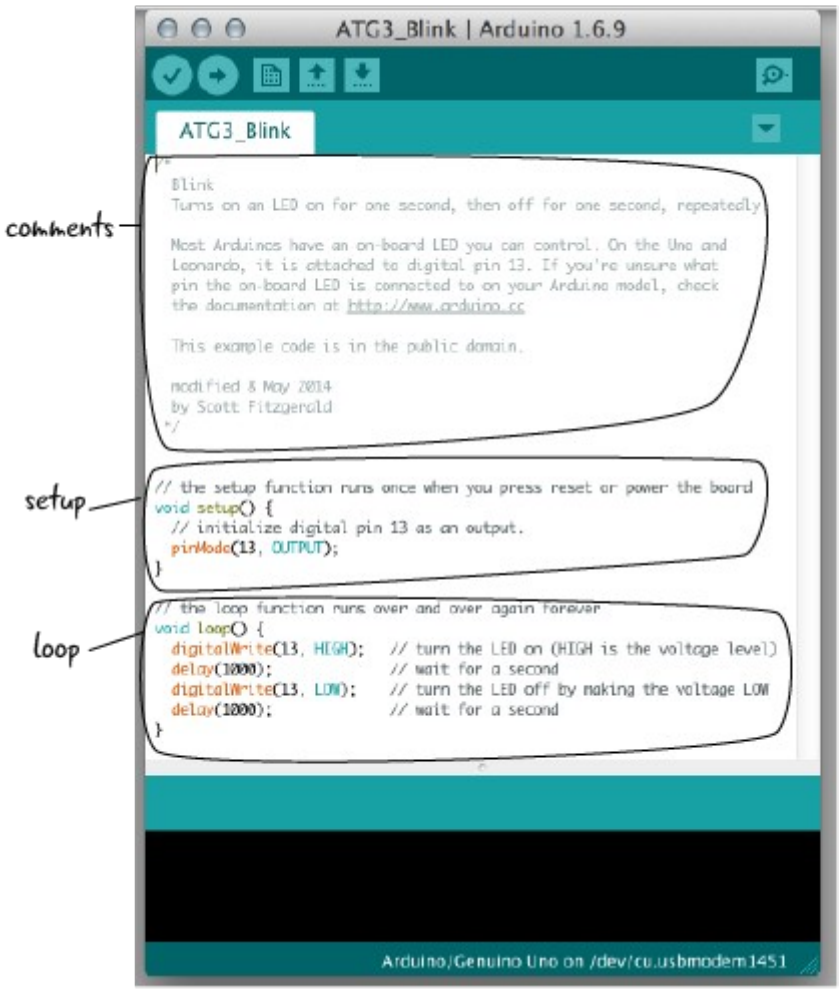
Low Power ARDF TX v1.0 “Wabbit”

You will find it under

File→ Examples→ Basics!



And we get something like this.



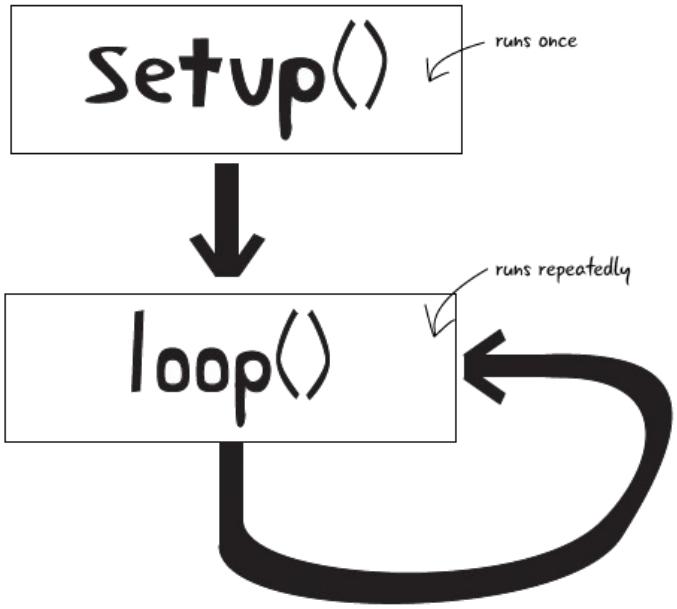
Don't stress out trying to understand this code-- we'll walk through it in detail in the coming pages.

It says digitalWrite(13 .. in this picture, but in my example it is digitalWrite(LED_BUILTIN ?

Yes, the examples can have small differences between version. The important part is to distinguish between setup and loop, and to see pinMode, digitalWrite and delay.

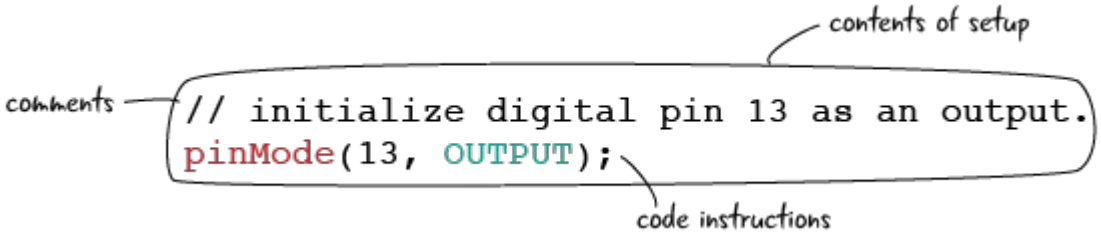
There are three parts to explain:

- comments: every text between "/*" and "*/" is a comment, and computer will not try to convert it into machine code.
  You can also write "// comment", if your comment is in one line
- setup and loop are two functions – setup is run once, at the beginning of microcontroller chip run – on power up or after doing a RESET. Loop just runs repeatedly – when it finishes it starts again.



Setup and loop are two functions – and they will run everything between the curly braces ("{" and "}").

Setup has this code within it:



Here we have a command – pinMode,
parameters to the command (13, OUTPUT),
and a semicolon ";" as each command must end with it (like a period at the end of a sentence.

The command pinMode sets what mode the line Digital 13 – it can be INPUT, INPUT_PULLUP (a special case for you to learn about) or OUTPUT. As the code was supposed to blink a LED, then the line should be set to OUTPUT.

Loop has this code (remember, it runs and runs and never ends):

Loop function from the Blink sketch

```
void loop() {
digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
delay(1000);              // wait for a second
digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
delay(1000);              // wait for a second
}
```

The functions here are digitalWrite and delay:

- digitalWrite is used like: digitalWrite(PIN, VALUE), where PIN is digital pin which you want to set either to HIGH voltage VALUE (so output of 5V) or LOW voltage VALUE (output of 0V).
- delay is used like delay(DELAY_IN_MS). So value of 1000 in the example above means that the program will stop (delay) for 1000ms = 1s before continuing.

So this loop function writes a HIGH voltage to PIN 13, waits for a second, writed a LOW voltage to same pin, waits for a second, and then it goes to the beginning, again with HIGH voltage....

If we tried this program in our Wabbits, it would do nothing. The reason is that pin Digital 13 is not connected to our LED1!

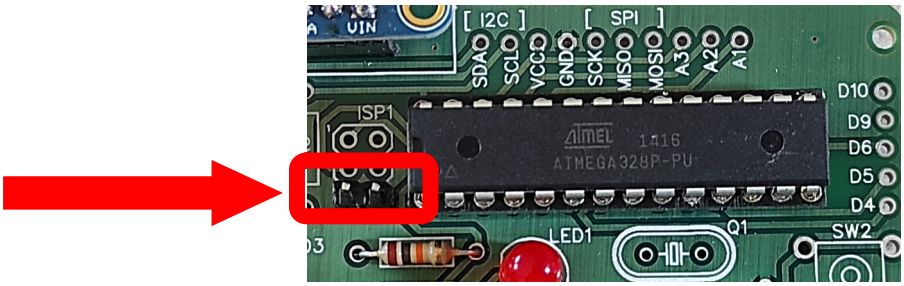
Check the schematic for LED1, and set the apropriate value in digitalWrite!

I'm unsure what's the correct digital pin for LED1, can you tell me?
Yes, the instructor could tell you (and probably will), but please, check the schematic.

Now, let's try to program the Wabbit:

click the Upload button  and look at the message bar if the upload succeded.

If upload didn't succed, try resetting your Wabbit after the upload button is pressed. This is done buy placing a screwdriver or a paperclip between the RESET and GND pins on the small two header next to the microcontroller.

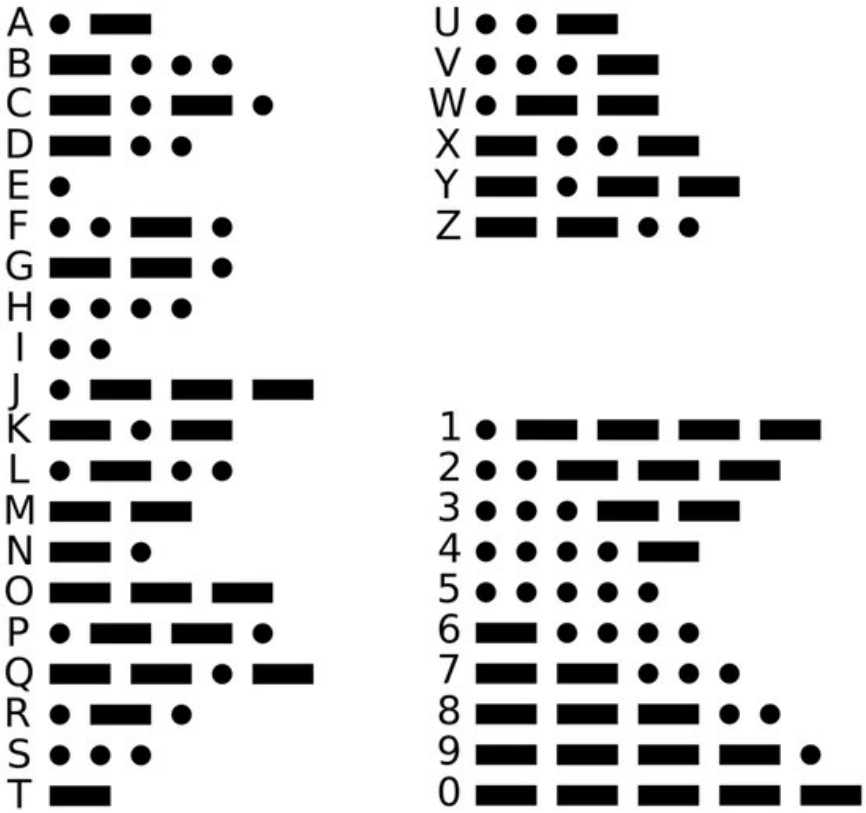


CONGRATULATIONS, YOUR WABBIT IS ALIVE!

Let’s experiment:

1. change the delay of only one of the HIGH and LOW parts, and upload new code! What do you observe? Can you explain it?

2. try to blink word “MORE” in MORSE code on LED1:



These are the timing rules for Morse code:

a) The length of a dot is 1 time unit.

b) A dash is 3 time units.

c) The space between symbols (dots and dashes) of the same letter is 1 time unit.

d) The space between letters is 3 time units.

e) The space between words is 7 time units.

Now it’s time to send this Morse code to the world!

Disconnect the USB-TTL converter from the computer, and plug the Si5351 oscillator module.

Connect the USB-TTL converte back to the computer.

Install the EtherKit Si5351 library if you don’t have it in Examples:

1. Go into Tools → Manage Libraries
2. In the Filter search type in “EtherKit Si5351”
3. Install the module

(second option it to use a ZIP library, and choose Sketch → Include library → Add ZIP library).

Let’s look at the Examples → EtherKit Si5351 (all the way down) → Si5351_outputs example:

```
#include "si5351.h"
#include "Wire.h"

Si5351 si5351;

void setup()
{
  // Start serial and initialize the Si5351
  Serial.begin(57600);
  si5351.init(SI5351_CRYSTAL_LOAD_8PF, 0, 0);

  // Set CLK0 to output 14 MHz
  si5351.set_freq(1400000000ULL, SI5351_CLK0);

  // Enable clock fanout for the XO
  si5351.set_clock_fanout(SI5351_FANOUT_XO, 1);

  // Enable clock fanout for MS
  si5351.set_clock_fanout(SI5351_FANOUT_MS, 1);

  // Set CLK1 to output the XO signal
  si5351.set_clock_source(SI5351_CLK1, SI5351_CLK_SRC_XTAL);
  si5351.output_enable(SI5351_CLK1, 1);

  // Set CLK2 to mirror the MS0 (CLK0) output
  si5351.set_clock_source(SI5351_CLK2, SI5351_CLK_SRC_MS0);
  si5351.output_enable(SI5351_CLK2, 1);

  // Change CLK0 output to 10 MHz, observe how CLK2 also changes
  si5351.set_freq(1000000000ULL, SI5351_CLK0);

  si5351.update_status();
  delay(500);
}

void loop()
{
  // Read the Status Register and print it every 10 seconds
  si5351.update_status();
  Serial.print("  SYS_INIT: ");
  Serial.print(si5351.dev_status.SYS_INIT);
  Serial.print("  LOL_A: ");
  Serial.print(si5351.dev_status.LOL_A);

  delay(10000);
}
```

Additional libraries and a structure, it's needed - just copy it!

Important function, which sets the frequency of CLK0

Starts the output.
Output_enable(SI5351_CLK0, 1); turns it on,
and
Output_enable(SI5351_CLK0, 0); Turns it off.

THE REST OF THE LINES ARE NOT NEEDED.

Copy the needed lines to your blink sketch, set frequency of CLK0 within CW part of 80m bandplan (3500-3600kHz), and let your Wabbit sing.

CONGRATULATIONS, YOUR WABBIT IS NOW A NEW TRANSMITTER (OF QRM) IN THE WORLD!

What's next?

Check out the examples:

- Digital→ Button and Digital→ Debounce on how use the SW1
- Analog→ AnalogInput or Basics→ ReadAnalogVoltageon how to use the potentiometer.

I’ve checked the examples but I don’t have an idea what to do?

Potentiometer: change the speed of the transmitting, or change the frequency on which it transmits.

Button: change between two or more speeds of transmitting, or change the message all together, change the frequency of transmitting, etc…