# ESP32-VROOM

This ESP32 module is used for the countdown timer.

* It connects with the robot through the BLE module so it can communicate with it.
* It uses a LED matrix board to show a clock that counts down.
* It controls a LED light strip to show color animations.

Text

Description automatically generated

We include the “arduino-timer” library which we also use on the Arduino Mega and ESP32-WROVER. This allows us to run code in an interval or timeout without blocking the main “loop()” function.

We create a new instance on the timer variable using the “timer\_create\_default()” function.

We also define a global boolean called “bluetoothIsConnected” which is false at first. This variable will allow us throughout the rest of the code to know if we are currently connected to the robot using the BLE bluetooth device.

Text

Description automatically generated

These are all the other files that are used for this device.

Files are split up by modules to clarify the code, the order is important here since code is run in C++ from top to bottom.

Text

Description automatically generated

This code is run when the ESP32 starts.

We will begin a Serial on BAUD 9600 (this is essentially the port that the serial communication uses to talk to another device). This Serial allows use to write errors and logs to the Serial Monitor in the Arduino IDE.

We then start the bluetoothSetup, this is however done in a way that allows use to run the function on another CPU core. The ESP32-WROOM has 2 cores, and by using both of them we can run code separated from eachother in a non-blocking way.

And example:  
If the code running on core 0 has a delay, it will stop the code execution untill the delay is over. This however doesn’t influence the code on core 1, so even if core 0 has a delay, core 1 can still keep running its code without issues.

The Arduino runs all its code on core 1 by default, so we want to start the bluetoothSetup on core 0. This way we can dedicate the full core to the BLE module, and it can use delays and while loops (whatever it needs) to receive and send bluetooth commands without blocking the code that runs the countdown.

The “runOnCore();” accepts 3 parameters, the function to run on this core, a name for the task that it runs, and the core to run it on. This function is defined in “Multicore.h” and will be clarified in the documentation assigned to that file.

We also run the ledMatrixSetup() and ledStripSetup() functions to initialize the code needed to set these up.

Text

Description automatically generated with medium confidence

This is the main loop() function, and runs as fast as it can (on core 1, as explained before).

We will run the “timer.tick()” function so the timer knows how much time went by, and can run the tasks assigned to it on the right intervals or timeouts.

We then run the “runLedStrip()” function (defined in “LedStrip.h”), this will make sure the right color animation is outputted to the LED strip.

After that, we run the “runTimer()” function (defined in “LedMatrix.h”), which will start, stop or update the countdown timer when it needs to.