# Electronics design exercise

*Mikhail Sheverdyaev*

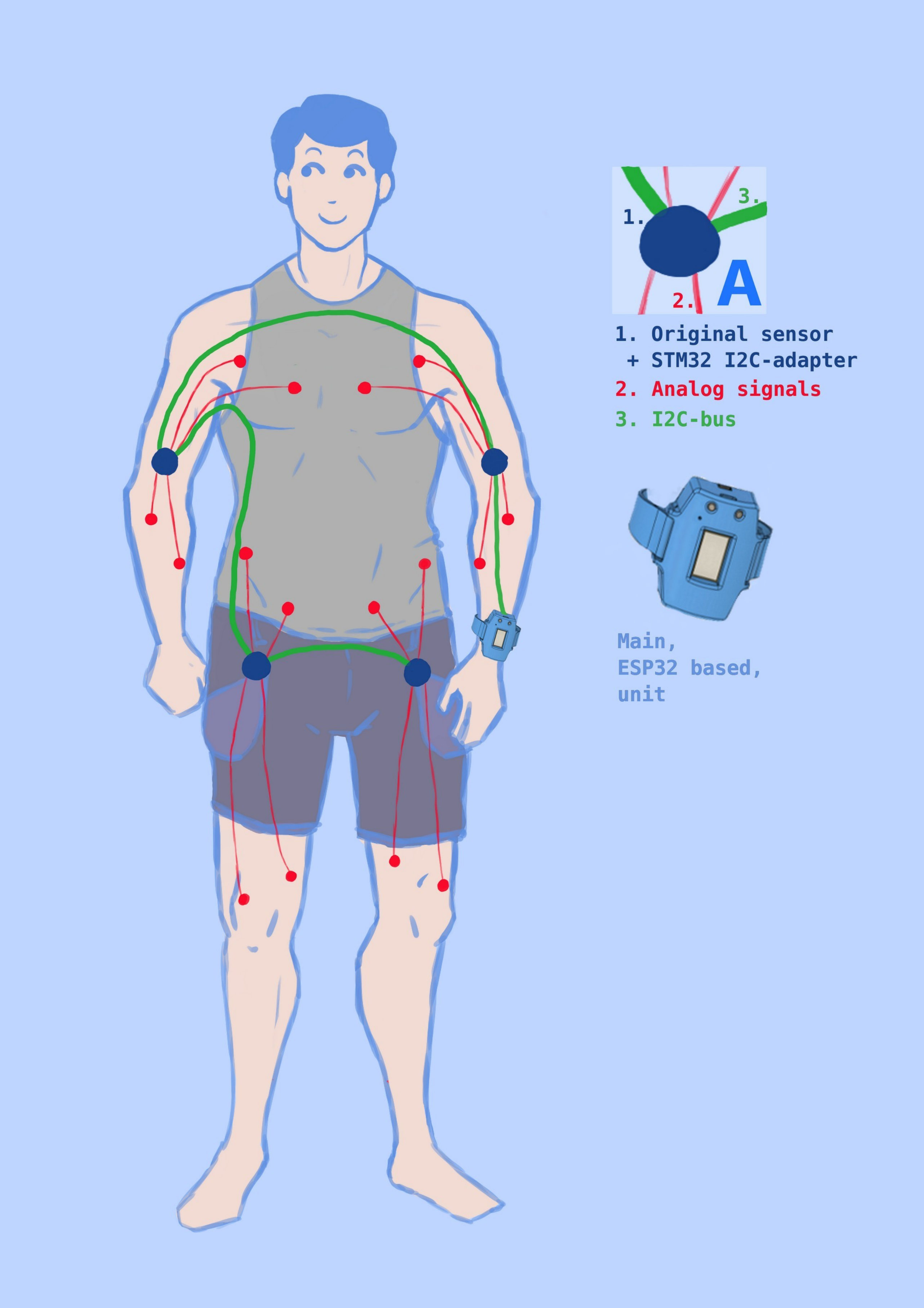
## General, and possible use-case scenarios assumptions:

- Since the device is designed for mounting it on the wrist, it has to be: lightweight and mobile, I.e. it should be able to work without wired connection to a power supply and / or a stationary computer, and have a sufficiently long battery life from a built-in source (for at least 4-6 hours).

- It is difficult to imagine a meaningful use of 16 analog sensors concentrated on a small area of the wrist, so I would assume that the device will be used as; a wearable electrocardiograph, pulse meter, blood saturation measuring or, monitoring the electric-activity of other muscles in the human body (when training athletes etc). That is, in this use-scenario, the sensors can be located from each other (and / or from the main board) at distance up to 2-3 meters.

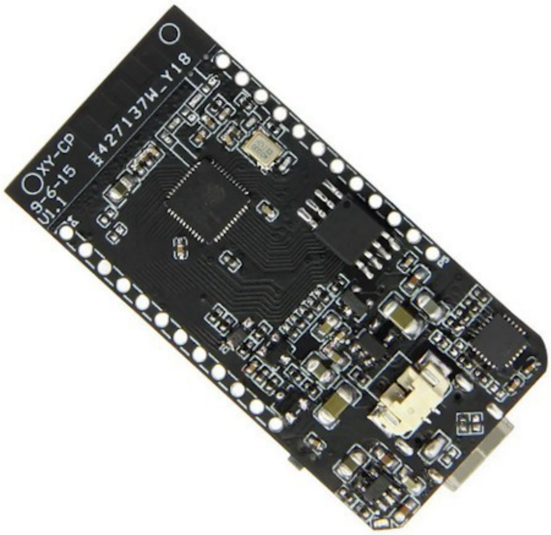
- The sensor's HW interface, how it is described here, looks very "specific" (if not saying - archaic) and the development of the device solely for it will not contribute to its future-proofing. Thus, IMO, it would be reasonable to convert it to some more commonly used type (I2C for an instance) by mean of a simple adapter. That will be needed anyway, if taking into account the rather long analog signals traces, and, thus, the neediness to perform AD-conversion as close to the signal source as possible. Otherwise, we could not fit the signal's integrity and its noise-ratio requirements.

- To build the sensor adapter we can use either "discrete" I2C ADC + IO extenders ICs ( ADC121C021 + MAX7323 for an instance), or some of the tiny MCUs. Both of these approaches have their strengths and weaknesses, but within this exercise frame, I will limit myself to the MCU version, as for the most interesting from a technical point of view.


1.
3.
2.
A
1. Original sensor
+ STM32 I2C-adapter
2. Analog signals
3. I2C-bus
Main,
ESP32 based,
unit

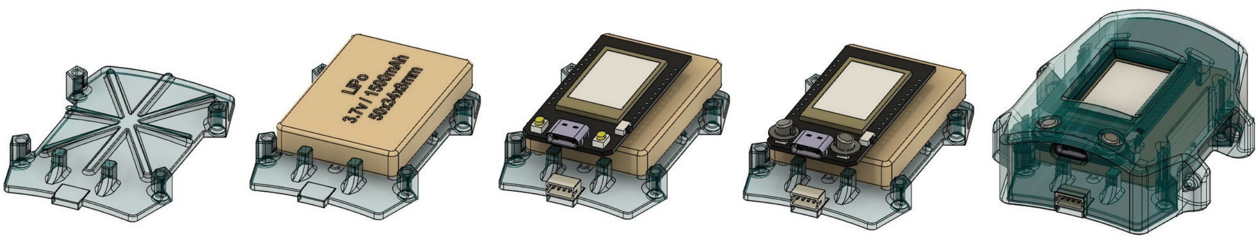
## Summarizing all of the above, Let's turn to the selection of the optimal components, circuitry and SW/FW solutions:

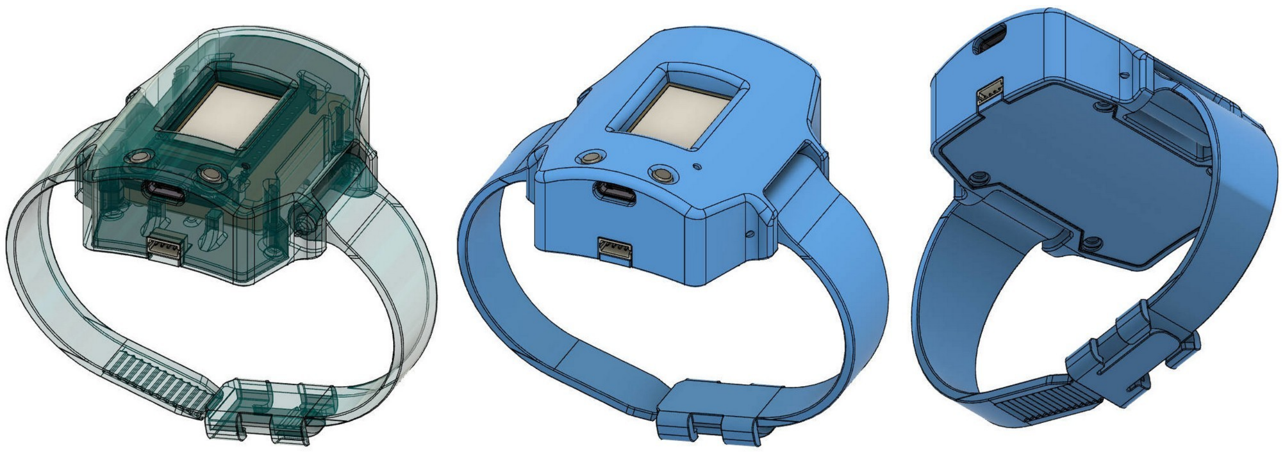
- For the main board my choice fell to (ESP32 based) TTGO-T-display, which has everything you need (WiFi, I2C, built-in LiPo charging circuitry and quite low power consumption), also, it does not require big modifications for using it in the project - In fact, we just need to expose its I2C-bus port outside, by soldering 4-pin p-1.25mm Molex 502386-470 connector to the corresponding pins on the board.

- For connection to the host computer, in the mobile mode, will be used WiFi and UART over USB-C port, in the wired one. There is also possibility to connect some remote sensor via Bluetooth ( both; WiFi and BT can work simultaneously there).

- This board will be placed to an ergonomic, Hand-Watches style body(/case), together with sufficiently capacious battery. Which can be charged via the same USB-C connector.

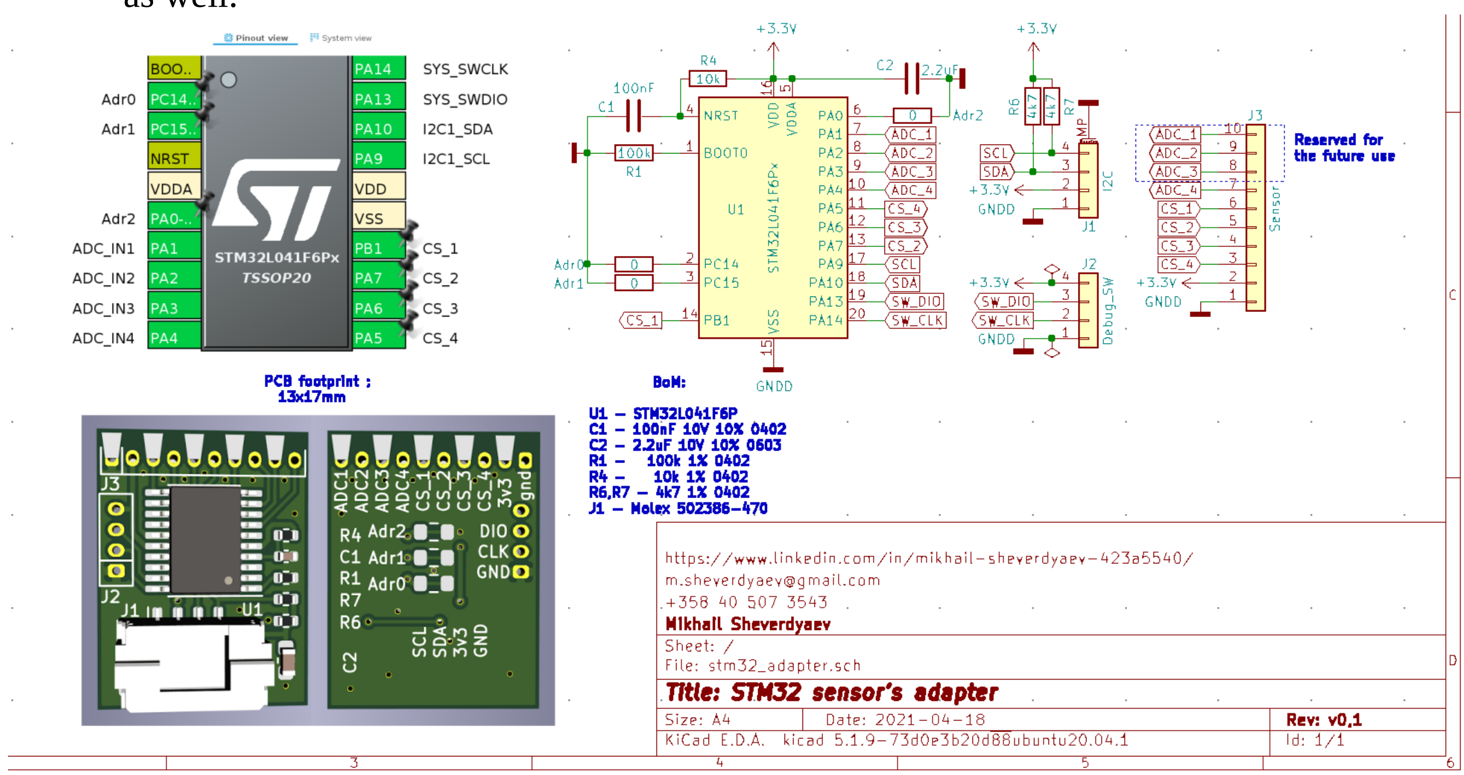




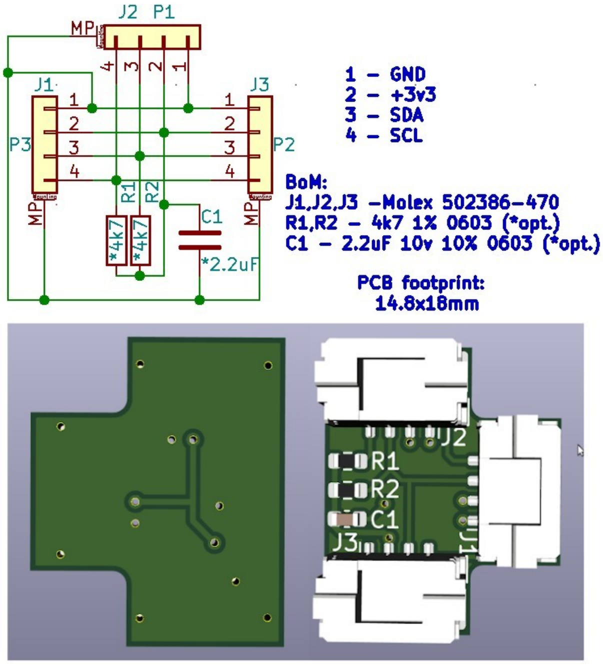
- For FW / SW development, it makes a sense to use MS VS-code in conjunction with PlatformIO plugin (and the Arduino framework). The FW will be based on FreeRTOS (tasks; Web-server, I2C-bus management, Service /Debug Terminal, Local Display, OTA, etc).

- The sensor adapter decided to be build on a tiny (TSSOP-20) STM32L041F6P MCU.

- Its has the same I2C-bus connector type as the main unit; i.e. Molex 502386-470. The question of choosing a proper connector to coup the adapter to a sensor remains open, due to the sensor's HW specifications absence. a place is reserved on the PCB for a 1.25mm pitch 7-pin (10 pin in fact) header that can fit rather the Molex 53048-0710(/1010) connector, or, just a bare wires can be soldered to there as well.



- In order to make the “sensor’s nodes” interconnections more  convenient and reliable, yet another simple “I2C-bus T- splitter” PCB has been designed:




- From the FW point of view, in order to minimize the I2C-bus traffic load, the adapter should  have an extended functionality, mean - it will manage the all connected sensor’s data acquisition in a semi-automated mode, by receiving just  one request command from main board (1+2x9+1 = 20 bits are needed) and sending all AD converted data during one bunch   reading transaction: 12x4/8x9 +1+9+1 = 65 bit.

- So, we can evolve the needed I2C bus speed/performance here: (20 + 65)x 100 x 4 = 34000 bit/sec. It means that, even if running the bus on its standard(low) 100kHz speed, we will still over-perform the requirement at almost 3 times.

- Same is about the AD conversion speed; the selected STM32 MCU ADC clocking 16Mhz, so it can perform 16/ (3+12) - a bit over than 1MS/s, while we would need just 400 S/s.

- Note: The adapter has 3 additional ADC-channels traced on  its PCB. Those, most probably, would be needed in the future – to avoid the analog signals multiplexing, as  it (looks like) realized in the “original sensor” design.

- For the adapter’s FW development, the "native" STM32CubeIDE (with HAL libraries: ADC, I2C, DMA) will be used.

## The system’s power-budget summary:

- Taking into account the total energy consumption; 4x50mA sensors + 30mA ESP32 + 4x10mA STM32 adapters = 270mA, a 1500mAh 3.7V battery would perfectly fits our autonomous working time requirement (5.5 hours).

- The selected battery has a 50x34mm footprint and there some other batteries with the same footprint in 1000 - 2200mAh capacity range are available on the market. They have just different height(?/thickness) - from 5 to 10mm and that has been taken into account during the case design.

## The Manufacturing Phases Plan:

1. Business / Use - cases studies.

2. HW concept feasibility studies, with using market available components(boards/ adapters). The main MCU/board selection and starting work on SW/FW architect plan.

3. First development HW prototype for building 1-2 kits; adapters/accessories, schematics/PCBs and mechanical parts 3D-models development. Ordering PCBs for an in-house assembling (JLCPCB.com or similar), 3D FDM case-printing just for its main geometry and the dimensions tolerance inspection. The "alpha"" SW/FW release is ready (just basic/core functionalities).

4. Product "release candidate" design . Manufacturing of up to 20-30 device kits: outsourced PCBa (PCBWAY.com or similar)and SLS/ HP-MJF (nylon PA12/PA2200) cases 3D-printing. Involving beta-testers for the product testing and validation.

5. "Lessons learned" analysis/summary and "Go/not to GO" decision taking.

6. The design optimization. Preparations for mass production and the customers support, or selling the device's design to some third party...

PS: The original case was designed specifically for this "exercise" in Auto Desk Fusion 360, - schematics and PCB layouts have made in KiCAD 5.1.9-73.

PSS: In case someone’s interested in getting this project’s source files (both, KiCAD’s and AD Fusion 360 ones), you can contact me by phone or email:

Mikhail Sheverdyaev
+358 40 507 3543
m.sheverdyaev@gmail.com