Kristel & Prabhav & Sara – 06/10/2025

MSc lab Autumn term – Wireless & Wired Sensor System

1. Aims and objectives

The aim of the laboratory is to gain experience and skills in building and testing sensors and sensor systems that include sensor nodes, wired and wireless communication and data analysis.

Tasks:

- Task 1: Design, fabricate (PCB) and test two autonomous sensor nodes (PPG & accelerometer).
- Task 2: Assemble an environment sensor system with commercial sensor boards and package it (3D print enclosure) to form another sensor node and test at different locations.
- Additional task when time allows: Use all the available sensors to devise the maximal number of approaches to measure the height of the EEE building and report on their accuracy.

2. Organisation

Each week, starting from week 2, a 3-hr lab session is timetabled. Look at your personal timetable for the scheduling. During these timetabled sessions, instructions and explanations will be given and members of staff will be available to support you. **You are expected to attend these sessions and work in your group**.
The MSc lab is on level 1 of the EEE department building. You will have access to this lab outside of the timetabled sessions during department opening hours when there is space available (other MSc courses in the department use the same lab space and you should not use the lab when those happening).
This lab requires you to have a laptop. Laptops can be borrowed from the automatic laptop dispenser next to stores near the lower-level entrance to the building.

The lab will be carried out in **groups of 4** who need to work as a team by dividing up the work while keeping each team member fully informed of the approaches and solutions taken. In the end, everyone needs to know everything even though they might not have fully contributed to each aspect of the work.

The members of staff involved in the lab are: Prof. Kristel Fobelets (room 714), Dr. Prabhav Reddy and Dr. Sara Ghoreishizadeh. Technical support is provided by Mr. Amine Halimi and Mr. Danny Harvey.

For the environmental sensor, each group will be given a lab box that contains an ESP32 S3 microcontroller, sensor boards, wires, battery. Groups will also be supplied with one Picoscope that connected to your laptop can be used as a oscilloscope. Lab boxes and picoscope will be on loan from stores on level 1 (near main staircase).

All PCB and 3D print designs need to be submitted via the EEE Tech Services - Power Apps. If you do not have access, please contact Mr Amine Halimi. Before production, submitted designs will first be checked by the technical team.

3. Deliverables and assessments

Task 1 autonomous sensor nodes

To be submitted:

1. The PCBs for production – 5% marks weighting
2. The data sheet of the two sensor nodes – circuit schematic, PCB design and performance parameters. The data sheet should not be longer than 10 pages. One data sheet per lab group. Should be uploaded on Blackboard. – 45% marks weighting

To be demonstrated in the lab

3. Operation of own sensor – 10% marks weighting

Kristel & Prabhav & Sara – 06/10/2025

Task 2 Environmental sensor

1. To be demonstrated to lab assessors: 20% marks weighting.

The environment sensor system within its enclosure + demonstration of its operation during lab session.

2. To be submitted: 20% marks weighting.

A report on environmental sensing including a comparison between your sensor and a control and a short critical analysis of your readings on the impact on public health. The report should not be longer than 5 pages and should be uploaded to github together with the code, enclosure design and the data. Link to github page to be uploaded on blackboard. One github page per lab group.

Task 3 How many ways to measure the height of the EEE building and with what accuracy

If time allows – groups can devise as many ways as possible to use all their sensors to measure the height of the building and reflect on its potential for accuracy. The results to be presented to the whole class via a powerpoint presentation.

4. Tasks

4.1 Task 1: Wireless Sensor Node Design and Test

In this part of the laboratory, you will gain hands-on experience with the design and development of custom wireless sensor nodes. The focus will be on:

- Designing printed circuit boards (PCBs) from the ground up, including layout and routing.
- Implementing on-board power management strategies to ensure energy efficiency.
- Designing antenna layouts to minimise interference and optimise wireless performance.
- Creating a custom USB extension board to facilitate programming of the microcontroller unit (MCU), avoiding the need for USB connectors on the compact sensor boards.

Each sensor board will wirelessly transmit data via Bluetooth Low Energy (BLE).

To do:

- **Design and submit** PCBs for two fully autonomous sensor nodes for fabrication.
- **Develop embedded software** to acquire sensor data and transmit it wirelessly to a database.
- **Design and document** a test protocol to evaluate the performance of your sensor nodes.

Deliverables:

- **1A: Submit PCB design** on EEE Tech Services - Power Apps for production.
- **1B deliverable: The data sheet** of both sensor nodes containing – circuit schematic, PCB design and performance parameters. The report should not be longer than 10 pages. One report per lab group. Reports should be submitted on Blackboard.

The PCBs will be fabricated and assembled by JLCPCB. You must consider JLCPCB's manufacturing capabilities and component availability during the design process.
You are required to design two autonomous sensor nodes at the component level:

- One node must include an **accelerometer**.
- The other must include a **photoplethysmography (PPG) sensor**.

More information about the working principle behind the sensor nodes can be found in Appendix A.

4.1.1 Designing the nodes

We expect two students to work on one node and the other two on the other. Of importance is that the power management circuit and the BLE circuit and layout are the same for the two sensor nodes. Thus, these designs are shared.
There is no reason to make these nodes ultra compact. Safe layout and relatively wide metal tracks are a better strategy to make a working node. In a development, a compact node would only be designed in subsequent design stages for which we do not have time.

For your sensor nodes you **must** use the following chips:

MCU	ESP32-C3FH4
BLE	ANT2012LL00R2400A
PPG	MAX30102
Accelerometer	LIS2DE12
Battery management system	DW01A/TLV75509/TPS22917

To design a circuit using these chips to build an autonomous sensor node you **must read the datasheets** of the sensors. Application circuits and values for pull up resistors and decoupling capacitors are given in the datasheets. Adhering to these recommendations is essential for a working node.

Kristel & Prabhav & Sara – 06/10/2025

Autonomous sensor nodes require an MCU[1] and an antenna for wireless communication, as well as a battery with a battery management system. In this exercise, you will use the ESP32-C3FH4 as microcontroller (MCU). Click on the ESP32 link for the datasheet. A reference schematic that you will need to build around the ESP32 chip can be found by clicking here. The ESP32 is a family of System-on-Chip (SoC) controllers that have an integrated BLE (Bluetooth Low Energy) and WiFi transceiver, a lot of General-Purpose Input/Output (GPIO) pins, is capable of numerous transmission protocols and are generally easy to write firmware for, with a lot of online resources available. The C3 has a very small package footprint and can reduce its power significantly using its sleep modes.

BLE will be used as wireless communication protocol for the autonomous sensor nodes. A small compact low power chip antenna will be used - the ANT2012LL00R2400A. The PCB layout[2],[3] needs careful attention to limit interference between the hardware around the MCU and the antenna. Leave distance between the BLE antenna and the rest of the circuit and ensure good grounding of the antenna area via vias to ground. The distance is also defined in terms of matching impedance thus again reading the datasheet is important to obtain this information.

The PPG sensor uses the MAX30102 chip and the accelerometer uses the LIS2DE12. Click on the links to be redirected to the datasheet of the sensors. In these datasheets you will find a typical application circuit that identifies the peripherals that are required. Note that the PPG sensor communicates via I^2C (Inter-Integrated Circuit) with the on-board MCU while the accelerometer chip can use either I^2C or SPI (Serial Peripheral Interface). A comparison between I2C and SPI is given in appendix B.

A Battery Management System (BMS) is an electronic control system that manages rechargeable battery packs by monitoring their condition, controlling their operation, and ensuring safe performance. For lithium-ion batteries specifically, the BMS serves as a critical safety component that prevents dangerous conditions while optimizing battery performance. You must implement a BMS on your autonomous boards. The BMS prevents the battery from overcharging, over-discharging, and overcurrent, which can damage the battery or reduce its lifespan. It monitors and controls temperature to avoid overheating, tracks battery health, charge level, and remaining capacity. It optimizes energy usage to extend battery life. Batteries do not supply a constant voltage under load, therefore an LDO (low dropout regulator) needs to be used to keep the voltage stable and step down the battery voltage to the level required by the different components on the board. The schematics of the BMS with a switch is given in Fig. 1. You can use this circuit for your BMS.

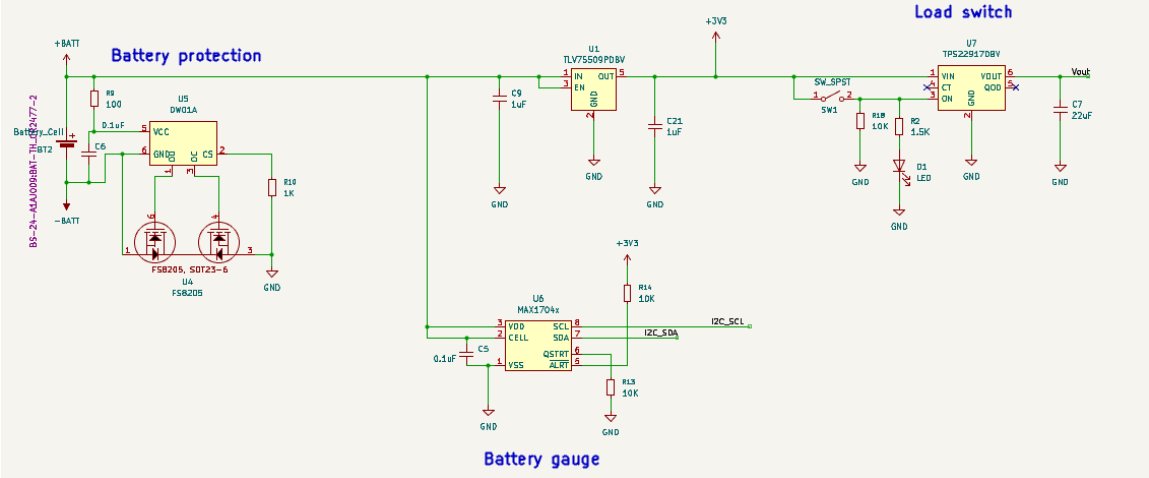


Figure 1: Battery management circuit built around the DW01A (designed by Amine Halimi) that can be used with the coin cell battery. This circuit also has a low dropout regulator and a switch. When programming the node, ensure the switch is off and no battery is present.

[1] Microcontroller unit

[2] Chip Antenna Layout Tips for BLE, WiFi & Zigbee | Johanson Tech

[3] AN91445 Antenna Design and RF Layout Guidelines

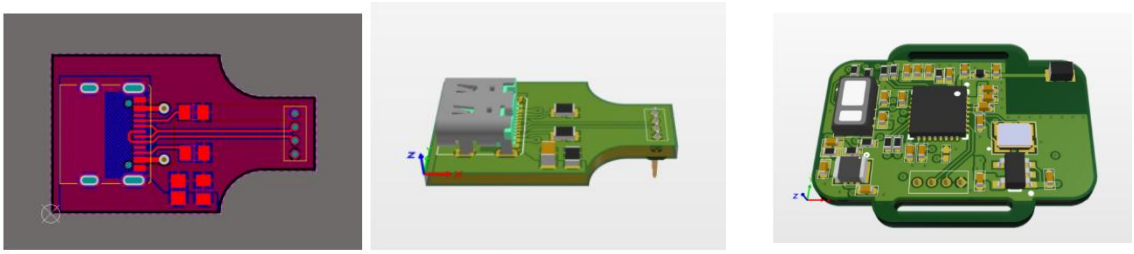
The battery charging circuit will not be implemented in the sensor node for health and safety reasons and also to keep the boards compact. A battery charging PCB to charge you battery safely will be supplied.

Once you have your circuit designs you can use easyEDA, kiCAD, Altium Designer[4] to implement your schematics and design the routings for PCBs. Note that the PCBs will be assembled by JLCPCB and thus, it is your responsibility to ensure all components are in stock. The easiest way to achieve this is using easyEDA where JLCPC-stocked components can be directly chosen.

Some possible free online learning materials:

EasyEDA	4 Layer PCB Layout Using EasyEDA Step By Step Tutorial | RootSaid
KiCAD	Designing a 4-Layer IoT Development Board with KiCad 9 - Tech Explorations
Altium	How to Design and Fabricate the Best 4 Layer PCB Stackup with Altium Designer

In your PCB design you must consider the method to program the ESP32 chip. This chip can be programmed directly via Universal Serial Bus (USB). You can choose to put the USB connector and peripherals onto your sensor board. This will increase the overall size of the sensor node but gives reliable connections. This can be avoided by separating the programming hardware from the node. A small USB extension board can be made with a 1.27 mm pitch 4-pin connector. Thus, on the sensor nodes a receptacle needs to be defined that connects the USB extension board to the MCU for on the boards for programming. While this gives more compact nodes and the extension board can be used for both board, connection is less reliable. An implementation of the extension board and the receptable is given in Fig. 2. Fig. 3 gives the schematics of the extension board.



(a) Layout (b) 3D View (c) PPG sensor node with receptable

Figure 2: Custom-made USB-to-sensor node connector board for programming the MCU on the node. (a) the PCB layout, (b) a 3D rendering and (c) the receptable (holes) on the PPG sensor node.

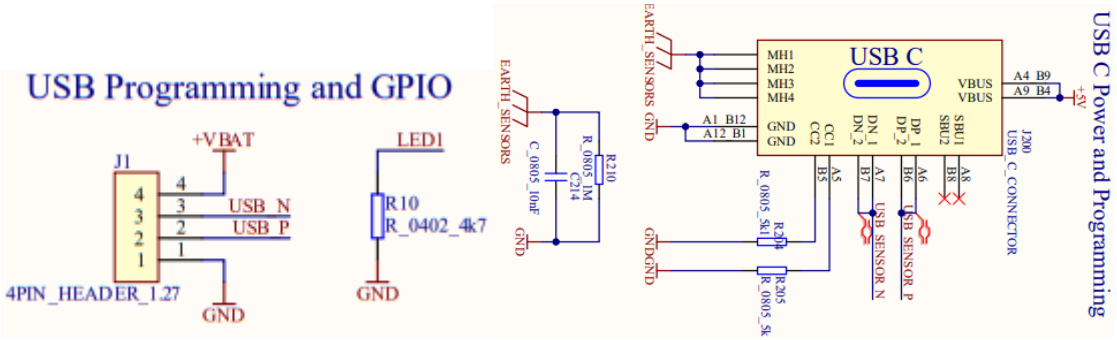


Figure 3: left the schematic for the receptacle on the sensor node. Right: the schematic for the extension board.

Silk screen: Use text on the silk screen to identify the different sections of your PCB. This is essential for troubleshooting, if required. Also, at the back of the PCB ensure you write Imperial College, the PCB version number and text identifying your lab group such that upon return the right PCB is handled over by stores. When your PCB is ready (before the deadline) you need to submit them via the EEE Tech Services - Power Apps. Your

[4] Also available via Imperial College London ICT software hub

designs will be checked for obvious flaws and if found will be returned to you with advice. Note that this is not a rigorous check of whether your design will work. When approved, your design will be sent by stores for fabrication in JCLPCB. You can now collect your environmental sensors box from stores on level 1.

4.1.2 Communication architecture

While you wait for the PCBs to be returned, you will be given home-built sensor nodes[5] that can be used for implementing the firmware for the sensor nodes and the wireless communication protocol. Note that while the specific libraries and commands for the sensors are different for the 2 boards, the wireless communication protocol is the same and can be shared within the lab group.

There are multiple ways to implement the software on the ESP32-C3 MCU, Arduino IDE, PlatformIO or ESP-IDF (amongst others). An overview of some features is given in Table I.

Table I: comparison between different platforms to implement the firmware

Feature	**Arduino IDE**	**PlatformIO**	**ESP-IDF (not supported)**
Ease of Use	✅ ✅ ✅	✅ ✅	✅
BLE Support	✅	✅ ✅	✅ ✅ ✅
Wi-Fi Support	✅	✅ ✅	✅ ✅ ✅
Power Management	Limited	✅	✅ ✅ ✅
Project Structure	Basic	Modular	Advanced
Debugging	Basic	Good	Excellent

Using Arduino IDE with ESP32-C3

1. Download and install Arduino IDE – link.
2. Install ESP32 Board Support
   - Open Arduino IDE.
   - Go to File > Preferences.
   - In Additional Board Manager URLs, add: https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json
   - Then go to Tools > Board > Boards Manager, search for ESP32, and install the package by Espressif.
3. Select the Correct Board
   - Go to Tools > Board and choose: ESP32C3 Dev Module
4. program in Arduino environment

250+ ESP32 Projects, Tutorials and Guides with Arduino IDE | Random Nerd Tutorials

Wired communication support:

- I2C: via Wire.h
- SPI: via SPI.h
- UART: via Serial, Serial1, etc.

Wireless Communication Support:

- BLE: Use the NimBLE-Arduino library (lightweight and efficient).
- Wi-Fi: Use WiFi.h for connecting to networks and sending data (e.g., via HTTP or MQTT).

PlatformIO Setup for ESP32-C3 Development

1. Install Visual Studio Code
   - Download from https://code.visualstudio.com & install
2. Install PlatformIO Extension
   - Open Visual Studio Code.

[5] If you used other chips than those proposed on p3 then this step cannot be done. You will need to talk to K. Fobelets to be given permission to proceed to the next step in the lab – the environmental sensor. This might mean that you might be late for the deliverable related to the data sheet.

- Go to the Extensions tab (Ctrl+Shift+X).
- Search for "PlatformIO IDE".
- Click Install.

3. Create a New Project

- Click the alien head icon (👽) in the left sidebar to open PlatformIO Home.
- Click "New Project".
- Fill in:
    1. Project name (e.g., ESP32C3_AccNode)
    2. Board: Select Espressif ESP32-C3 Dev Module
    3. Framework: Choose Arduino or ESP-IDF.
- Click Finish. PlatformIO will generate a project folder with src, lib, and platformio.ini.

4. Write Your Code

- Place your main code in src/main.cpp.
- Use libraries in platformio.ini
- Tutorials and Examples — PlatformIO latest documentation

It would be good to keep in mind that the wireless protocol you develop for these sensor nodes can also be used for the environmental node you will need to build in the next task for the lab. Thus, ensure that your approach is transferrable.

A possible architecture of the sensor system could be as shown in Fig. 4.

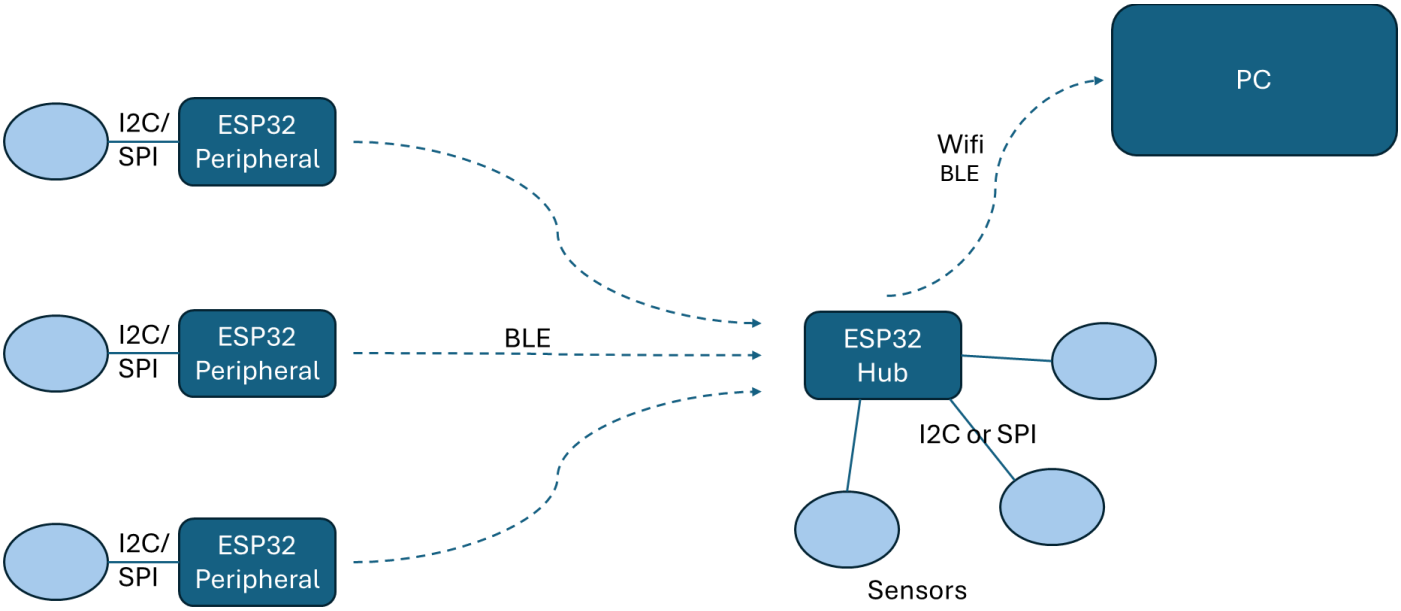


Figure 4. A general architecture of the sensor system that can be built.

As seen in Fig. 4, the sensor system has an ESP32 hub which collects data from sensors which are physically connected to it using I2C or SPI (e.g. the environmental sensor in Task 2). In addition, the hub can also collect data from other sensor nodes. This would be via wireless communication using BLE for the autonomous sensor nodes. The hub then arranges the data and sends it to a PC where it is stored in a database. The connection to a PC is via Wi-Fi. Instead of sending the data to the PC, you can store all the data on an SD card attached to the sensor hub – this makes sense if the PC is not available. Alternatively, you could use nRF Connect to a smart phone to collect the data.

Description related to Fig. 4:

1. ESP32 Peripheral: The peripheral needs to collect data from a sensor and sends it to the hub. It is usual to configure this as the BLE server if you are using BLE.

2. ESP32 Hub: The hub will receive data from the peripherals. If you are using BLE, the hub is usually configured as a BLE client. It may need to collect data via I2C or SPI from other sensors. In addition, the hub will have to configured as a web server. At periodic intervals, the hub pushes data which is read by a code on the PC and stored in the database.
3. PC: You will need to listen for server-sent events and store the data in a database. We have worked with SQLite databases (Using sqlite3 library) but many other database and python libraries exist.
4. We have tested both home Wi-Fi (at home and through the phone hotspot) and enterprise Wi-Fi (ImperialWPA) and have been able to get the ESP32 to connect to them.

Helpful resources:

1. ESP32 BLE Server and Client (Bluetooth Low Energy) | Random Nerd Tutorials - The tutorial shows how to send data between two ESP32 devices using BLE.
2. ESP32 Web Server with BME680 - Weather Station | Random Nerd Tutorials - The tutorial collects data from a BME680 sensor and displays it on a webpage.
3. ESP32: ESP-NOW and Wi-Fi Web Server Dashboard (Arduino) | Random Nerd Tutorials - The tutorial shows how sensor data collected by a peripheral is sent to a hub using ESP-NOW. The hub displays the data on a webpage.
4. sseclient · PyPI - The sseclient library in Python is used to handle Server-Sent Events (SSE), which allow servers to push real-time updates to clients over HTTP. It provides a simple way to consume event streams from web servers.
5. NimBLE-Arduino - The BLE library is quite large and we found that along with the HTTP Client library was exceeding the program memory of the ESP32. Because of this we replaced the BLE library with NimBLE, which is a light-weight version of the ArduinoBLE library. We found the migration guide (from ArduinoBLE to NimBLE) very helpful.

4.1.3 Testing

The deliverables of this section

- Design PCBs for two fully autonomous sensor boards and submit for production at the latest by the deadline.
- Develop software to read out your sensors nodes and send the data wirelessly to a database.
- Devise a test protocol to analyse the performance of your autonomous sensor nodes.

When your sensor nodes are fabricated, implement the wireless communication protocol to save the data to an online database.

If your sensor nodes are faulty, you can use the department sensors nodes for further development. If you need to do this, then you need to report this in your datasheet. The datasheet should still report on your schematics and PCB design. The testing will be based on the departmental sensors.

Different decisions on how to collect the data will need to be made at this stage.

The sensor data must be **sampled** carefully to avoid inaccurate readings and excessive power consumption. Oversampling can be computationally and energy inefficient, while under-sampling (sampling frequency below the Nyquist criterion $< 2 \times f_{max}$) will lead to data loss and aliasing. Thus, sampling rates should be tailored to the specific application—e.g., body temperature sensors may only need a few readings per minute.

Heart rates are normally not higher than 200 BPM for young fit people. A sampling rate of 100 – 250 Hz for the PPG is sufficient for consumer wearable but should be much higher (~ 1000 Hz) for research or medical applications where signal fidelity is critical. Commercial smart watches sample at 25–100 Hz.

Sampling rate for *accelerometers* is determined by its application. The sampling rate for the accelerometer depends on the application and will be limited by the chosen accelerometer technology. Some values are proposed in Table II.

Table II: Common sampling rates by use case.

Application	Typical Sampling Rate
Basic activity tracking	20–50 Hz
Pedometer / step counting	~100 Hz
Fall detection / motion events	100–400 Hz
Human activity recognition	50–100 Hz
Biomechanics / gait analysis	125–250 Hz
High-precision motion tracking	Up to 1000 Hz

The recorded data will contain noise and should be filtered for further data extraction.
Interference by noise sources should be considered. Typical noise sources are power grid interference (~50 Hz in the UK), electrical circuit noise (can be minimised through good PCB design) and body movement noise (e.g. breathing, muscle activity, friction) need to be filtered out. Digital filters are useful but computationally demanding, outsourcing them to data hubs can improve power efficiency if it doesn't increase transmission load. Analogue filtering will reduce on-MCU digital filtering demands but increase power load on the nodes and increases the node's footprint. Thus, a compromise will need to be made.

An overview of filters typically used in IoT systems that can be implemented on the MCU, from the very simple to the complex:

- Moving Average Filters: Simple smoothing technique to reduce random noise.
- Polynomial fitting/detrending: fits a curve to the baseline and subtract it.
- Bandpass filters: remove high frequency noise and low frequency motion noise. Types:
    - Finite Impulse Response (FIR): Stable and linear phase; good for precise filtering.
    - Infinite Impulse Response (IIR): More efficient but can introduce phase distortion.
- Notch filters: target and remove specific frequencies like the 50 Hz power line interference.
- Kalman Filters: Advanced filters used for sensor fusion and motion tracking (e.g., combining accelerometer and gyroscope data).
- Adaptive Filters: Adjust their parameters based on signal characteristics (e.g., active noise cancellation).

Digital filters on the MCU must be lightweight, fast for real-time filtering and should preserve signal integrity. Once you have designed and implemented your readout protocol you need to test the autonomous sensor nodes.

Tests that must be carried out

Please refer to your learning in the Sensors module to find the important parameters that need to be extracted from the sensor.

Accelerometer Calibration and Step Detection Exercise

Before using your accelerometer for motion analysis, it is important to calibrate it. This is because the sensor may not be perfectly aligned with the true zero of the x, y, and z axes. A simple calibration method is available by clicking on here. Perform the calibration and report your results using crosshair circle plots to illustrate the sensor's alignment.

Kristel & Prabhav & Sara – 06/10/2025

To test your calibrated accelerometer, walk through the Level 1 corridor while manually counting your steps. Simultaneously, record acceleration data using your sensor. You will need to implement a step detection algorithm to extract the number of steps from the recorded data.

Plot the number of steps detected by your sensor against the number of steps you counted manually. You will need to do this multiple times and by different members in your group to allow for some statistical analysis. Analyse the accuracy of your step detection, discuss any discrepancies, and reflect on possible sources of error or improvements to your algorithm.

PPG Sensor Setup and Heart Rate Measurement

PPG sensors should be placed on areas of the body where blood vessels are close to the skin surface, such as the wrist or fingertip. Because PPG readings are sensitive to pressure, pressing your finger directly onto the sensor is not a reliable method for measuring heart rate. Instead, mount the sensor on the wrist using a strap or fixture that maintains consistent contact pressure—firm enough for good signal quality but not so tight that it restricts blood flow. To test your PPG sensor, measure your heart rate using the sensor and simultaneously by placing your index and middle fingers on the carotid artery in your neck.

Repeat the measurements at different heart rates. You can increase your heart rate through light activity such as walking, running, or jumping. However, for accurate recordings, we recommend taking measurements while seated and still, to minimize motion artifacts and noise. Take these measurements multiple times and by different people in your group to allow for some statistical analysis.

To be submitted:
The data sheet of both your sensors containing – circuit schematic, PCB design and performance parameters. The report should not be longer than 10 pages. One report per lab group. Submit on blackboard before the deadline.

4.2 Task 2: Environmental sensing

The aim of this section is to make an environmental sensor system based on commercial sensor boards and gather data from different locations in London.

The deliverables of this task are:

- Develop the code to read all the sensor nodes, display them on the OLED, save the data to SD card and send them wirelessly to a database.
- Design a compact PCB to connect all sensor boards to the MCU without wiring.
- Design and 3D print a box to contain your environmental sensor for safe deployment using the battery pack.
- Gather environmental data with your sensor board at different locations and compare to available data.

To do:

- **Code** the ESP32 S3 feather MCU either on PlatformIO or using ArduinoIDE. The coding can be developed with the sensor board wired via I2C or SPI.
- **Design** a PCB that assembles all sensors around the MCU in a compact way, suitable for packaging.
- **Design** an enclosure using SolidWorks and 3D printing to contain your sensors PCB.
- **Use** your environmental sensor to monitor lab and to take readings outside. Compare recordings with available commercial data.

Organisation

There is a lot of work to be done in this section and allocating different tasks to different members of the group will be essential. For instance, two people can develop the readout software, two people do the PCB and enclosure design. These groups of two do not need to be the same as for the autonomous sensor designs. Communication between the groups is essential. Note that wireless data transfer was already developed and can be re-used.

Deliverables:

- **Submit PCB design** on EEE Tech Services - Power Apps for production.
- **Submit github address.** Github page should contain PCB design, enclosure design, code, measured data of own environmental sensor and other commercial sensor and a report including pictures of the as well a critical analysis. Github link should be submitted on Blackboard.

The box

The list of items that will be given to each group is given in Table III. These can be obtained from stores level 1.

Table III: components and equipment given to each group by Stores

Part number	Description	Website link
1528-5477-ND	Adafruit ESP32-S3 Feather with 4MB Flash 2MB PSRAM - STEMMA QT / Qwiic	ESP32-S3 Feather
1528-5046-ND	Adafruit BME688 - Temperature, Humidity, Pressure and Gas Sensor - STEMMA QT	BME688
1528-4415-ND	Adafruit MINI GPS PA1010D STEMMA QT	Mini GPS
1528-5425-ND	STEMMAQT VL53L4CX TIME OF FLIGHT	Time of Flight Distance Sensor
1528-1462-ND	MICROSD CARD BREAKOUT 5V OR 3V	MicroSD card
1528-1220-ND	Monochrome 0.96" 128x64 OLED Graphic Display - STEMMA QT	OLED display
RS	PicoScope 2405A	PICOSCOPE 2405A datasheet
	Rechargeable coin-cell batteries for autonomous sensor nodes.	

2987-DH-20UE0062-NH-ND	C-TYPE USB cable	
1528-4399-ND	STEMMA QT / Qwiic JST SH 4-Pin Cable - 50mm Long	
	Pinheader cables	

Home-made autonomous sensor nodes will be made available when own designs are submitted.

N/A	Home developed accelerometer autonomous sensor node	N/A
N/A	Home developed PPG autonomous sensor node	N/A

At the end of the Autumn term all items need to be returned in the box to stores.

4.2.1 Introduction to Environmental Sensing

Environmental sensing refers to the use of sensors and data acquisition systems to monitor physical, chemical, and biological conditions in the surrounding environment. These sensors can measure variables such as air quality, temperature, humidity, noise levels, radiation, and pollutant concentrations, among others. In urban settings like London, environmental sensing plays a crucial role in understanding and managing the impact of human activity on public health and ecological sustainability. Distributed sensor networks provide real-time, high-resolution data that can inform policy decisions, support scientific research, and empower communities to act. Breathe London is a city-wide air quality monitoring network managed by Imperial College London and offers real-time and historical data from over 400 sensor nodes. The data includes PM_1, $PM_{2.5}$, PM_{10}, NO_2, O_3, temperature, humidity and pressure and can be accessed via interactive maps, CSV/Excel downloads or via an API for developers.

Why is environmental sensing important?

1. Public Health Protection. Monitoring air pollutants like NO_2 and $PM_{2.5}$ helps identify health risks and supports interventions to reduce exposure, especially for vulnerable populations.
2. Climate and Urban Planning. Temperature and humidity data contribute to climate resilience strategies, such as designing green infrastructure and managing heat islands.
3. Regulatory Compliance. Continuous sensing ensures that environmental standards are met and helps detect violations early.
4. Citizen Engagement. Open access to environmental data fosters transparency and encourages public participation in sustainability efforts.
5. Smart City Development. Integrating environmental sensors into urban systems enables adaptive responses to changing conditions, enhancing liveability and resource efficiency.

4.2.2 Building your environmental sensor system

Although none of the sensor boards can monitor NO_2 and $PM_{2.5}$ they can observe some of the commonly monitored parameters. The BME688 will give an overall VOC resistance value but does not distinguish between gasses and alcohols. It can be used to calculate an air quality index.

This node will be relatively large as it will contain multiple sensors as well as data display and storage devices and will use an ESP32 S3 Feather MCU development board. The boards can use I2C or SPI communication protocols.

You have been given multiple commercial breakout boards: an environment sensor BME, a GPS module and a time-of-flight module. You have also been given an OLED display and a microSD card reader. You are strongly advised to read up on how these sensors work. All these boards need to be connected to the ESP32 development board. All the boards are capable of I2C communication, apart from the SD card reader that

operates using the SPI communication protocol. Some of the boards have both I2C and SPI capability. Most boards, including the MCU are fitted with a qwiic/stemma QT connector that supports I2C and allows you to daisy chain your different boards to the MCU with the special wires supplied in your pack, as long as they have different addresses (see Fig. 5a). In Fig. 5b the qwiic connectors are not used and all the boards are wired to the MCU GPIO pins by either I2C or SPI using jumper headers. In Fig. 5c the boards are all connected via a custom-made PCB on which the routing between boards and MCU is implemented. This is surely the least messy, most compact and most robust implementation. You need to implement your environment sensor system on PCB as shown in Fig. 5c. The PCB in Fig. 5c is only for illustration purposes and should not be copied, there are more elegant solutions.

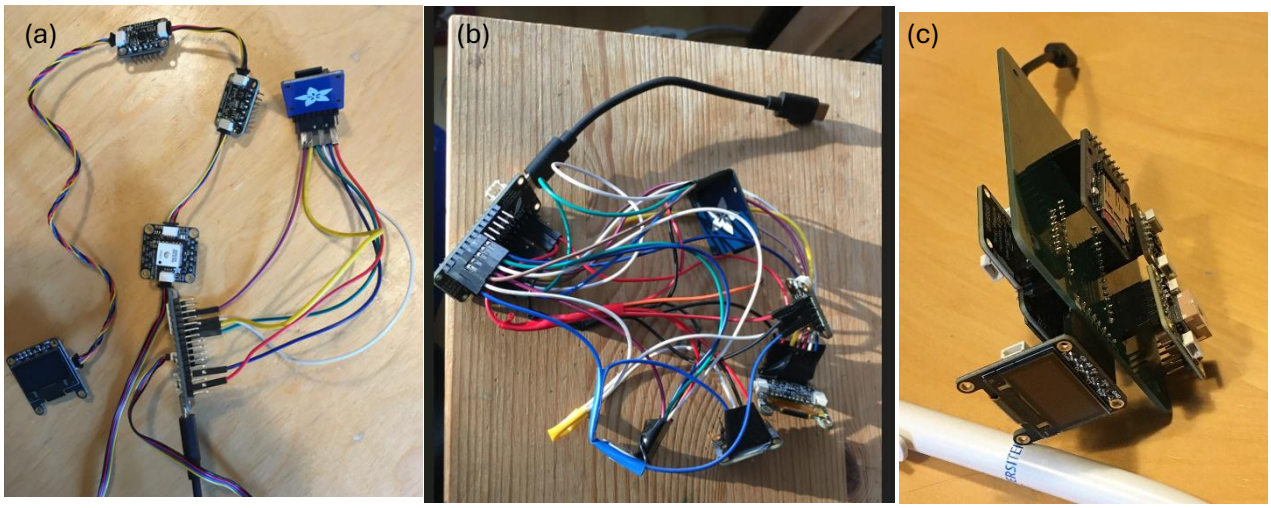


Figure 5: all boards are connected to the ESP32 MCU development board (a) using the qwiic connectors to daisy chain all possible boards, (b) using wires for I2C and SPI communication with the ESP, (c) using a custom-made board.

Different tasks to complete (divide the work amongst the team members).

A) Program the ESP32 to read out all sensors, display the readings on the OLED and save it to the SD card. This can be done by using Arduino IDE or PlatformIO or other proprietary software. This can be started from wired connections, but you must ensure that you can transfer your implementation to the system on PCB where are these wires are absent (cfr. Fig. 4c).
   Example code is available online for all commercial boards. See the Adafruit website for info as the boards are made by them. You will need to upload all the necessary libraries for the different Adafruit boards as well as the libraries for I2C and SPI communication. The library for the correct ESP32 development board also needs to be uploaded.
   It is probably best to take an approach of implementing each sensor separately and when it's working increase the number of sensors. Test by writing info to the serial monitor. This will need to be removed in the final implementation.
   The OLED can be a bit slow in starting up and you should take that into account in your programming. The OLED screen is too small to display all the sensor information in one go and thus you will need to implement a solution to show all data.
   Write csv files to the SD card reader. USB-microSD cards convertors to download the data to your computer can be bought if you do not have one. Use excel or MATLAB to read and analyse the data.
   The GPS board comes across as slow since in the beginning it needs to localise different satellites. Depending on where you are and what time of the day, this can be either fast or slow. With the available battery the last satellite position can be retained for fast start-up. The GPS will not work indoors.
B) Design a compact PCB to route all connections between boards and MCU. The boards will be checked by Amine Halimi and will be ordered from JLCPCB via stores. The sockets can be assembled by JLCPCB. Since an enclosure needs to be fabricated for this PCB and the power bank, you are strongly encouraged to add

the 3D CAD designs of all parts (commercial boards) you are using to the PCB design such that you can visualize the geometry of the design. CAD designs can be downloaded from different sources such as the Adafruit github page, SnapEDA, GrabCAD, UltraLibrarian, 3DContentCentral. In the absence of available CAD designs, you will need to design your own for which you can use SolidWorks that will also be used to design the enclosure. An explanation of SolidWorks can be found in Section 4.2.3.

C) Design and print a box to house your PCB. Ensure that openings are made to allow access to USB, SD card, etc. Remember the time-of-flight sensor is optical. (see section 4.2.3). Designs will need to be submitted via: EEE Tech Services - Power Apps and will then be sent by Amine Halimi to the printers on level 1.

D) Use the battery pack and the boxed environmental sensor system to take measurements at different locations inside and outside the department and compare the readings from your sensors to known data. An atmospheric sensor will be made available in the lab – a Lascar EasyLog EL-WEM. You can download the EasyLog Cloud App to obtain the readings. When outside the comparative data can be found from the environment sensors in London and downloaded. These sensors have a lot more capabilities than your environmental sensor.

5.3 Design and 3D print enclosure

To protect your circuit while monitoring the environment in different places in and around College you need to design and print an enclosure to hold your PCB with all sensor boards connected as well as your battery pack. Most sensors cannot be fully covered by the enclosure as they need access to the environment. Thus, your enclosure will need to accommodate for this.

SOLIDWORKS is a 3D design software that is used to design all sorts of products. It is a great resource for teaching 3D mechanical CAD, design validation, and much more. You can get access to SOLIDWORKS via College ICT. Learning guides are available via the Solidsolutions website. Your print job will need to be submitted to the EEE Tech Services print queue. Before printing, Amine Halimi will check the feasibility of the print (but not check whether everything is correct for your particular PCB layout).

Some items to consider when designing an enclosure for an environmental sensor PCB that also holds a battery.

1. Initial Setup
   - Import accurate PCB dimensions (DXF, STEP, or manually measured)
   - Include all component placements and heights (OLED, sensors, connectors, battery)
   - Set correct units (mm or inches) and tolerances for 3D printing
2. Enclosure Geometry
   - Define overall enclosure dimensions with clearance for components and wiring
   - Add mounting standoffs for PCB (check hole sizes and spacing)
   - Include internal supports or compartments for battery pack
   - Design snap-fit features or screw bosses for assembly
3. Openings and Interfaces
   - OLED display window or cutout with bezel/lip
   - Time-of-flight sensor aperture or transparent window
   - SD card slot with finger clearance
   - BME680 vent holes or mesh for airflow
   - USB or power connector access
4. Thermal and Environmental Considerations
   - Ventilation holes or slots near heat-generating components
   - Avoid enclosing environmental sensors in sealed compartments
   - Consider printing material thermal properties
5. Cable and Component Clearance
   - Check clearance for headers, and connectors
   - Avoid sharp corners near cables or battery leads

- Include strain relief features if cables exit the enclosure

6. Assembly and Maintenance
   - Ensure easy insertion/removal of PCB and battery
   - Avoid tight fits that require force
   - Design for tool access if screws are used
   - Consider modularity for future upgrades
7. 3D Printing Preparation
   - Orient parts to minimize overhangs and supports
   - Add fillets/chamfers to reduce stress and improve print quality
   - Check wall thickness (≥1.2 mm)
   - Export STL with correct resolution and units
8. Labelling and Aesthetics
   - Add embossed or engraved labels for ports/sensors
   - Ensure intuitive orientation (e.g., display facing up)
   - Consider surface texture or branding elements

Tests that must be carried out:

1. Functioning of the environmental sensor system integrated in the board. Does the display work and give all the relevant information: Temp (^{0}C), Pressure (hPa), Humidity (%), Gas (kOhm), Distance (mm), latitude, longitude, altitude (m) and speed (m/s). Is the data saved to the SD card. Note that for the GPS to work, you need to go outside (when it doesn't rain). Take readings in front of the lab for the positioning only. Note that if your system does not work on PCB, then carry out the recordings with the wired system.
2. Demonstrate the enclosure fits around the sensor system and allows recordings as well as display of the recordings.
3. Test the system outside. For this you will need to send the data wirelessly to a database unless you take your laptop on your test round. Since the VOC sensor is not calibrated you have to calibrate it against a known environment. You can check the Breathe London website for areas with different levels of pollution and match their air quality index (AQI) with that extracted from your sensor readings. For this you will need to find an algorithm that calculates an AQI based on your sensor readings[6]. We expect that the different groups will take readings at different places. Record the place (name and GPS) where you take your readings.

To be demonstrated to lab assessors:
The environment sensor system within its enclosure + demonstration of its operation and comparison to the lab sensor data.
To be submitted:
The data of the environmental sensor experiment compared to its control and critically analysed for impact on public health. You must give the date and place for your measurements. The report should not be longer than 5 pages and should contact a link to github page with the code, enclosure design and the data. One report per lab group.

4.3 Task 3: Measuring the height of the EEE building

The additional task only needs to be done if time allows. This exercise requires creativity and out-of-the-box thinking. In this last task you need to use all possible sensors you made/assembled during the lab to measure the height of the EEE building. How you do it is up to you, be inventive.

[6] E.g. see the Bosch's BSEC library for advanced air quality index (IAQ) calculations

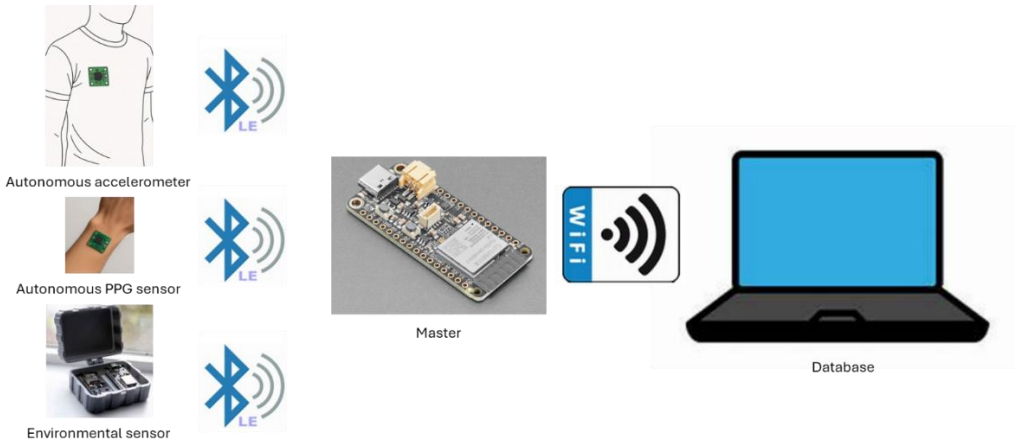


Figure 6: A possible architecture for transferring the sensor data of all sensor nodes to a computer to measure the height of the EEE building.

List all the approaches you can imagine, apply them to estimate the height of the EEE building and think of the pros and cons of the approach able to explain why certain approaches might give accurate/inaccurate results. The aim is not to have the most accurate measurement (that is a bonus) but to understand the limitations of the different sensors for the task given. The group with the most appropriate approaches will win the MSc Sensor System lab prize.

Deliverable: present your ideas and results to the whole class clearly communicating your understanding of the strengths and limitations of the approaches. This can be done with a Powerpoint presentation in the lab.

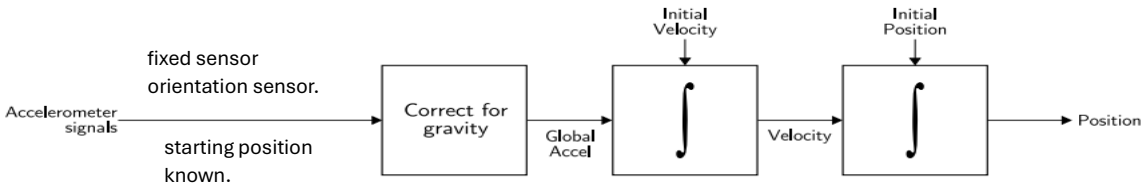
Appendix A: Brief description of the sensors

An accelerometer detects changes in velocity, *v* by measuring the force exerted on a small mass inside the device (Newton's second law).

$$\boldsymbol{F} = m \cdot \boldsymbol{a} = m \cdot \frac{d\boldsymbol{v}}{dt} \quad (1)$$

with ***F*** the force, *m* the mass, ***a*** the acceleration and ***v*** the velocity

The accelerometer used in the lab is based on MEMS (Micro-Electro-Mechanical-Systems) technology. A tiny proof mass is suspended by beams. When the device accelerates, the mass lags due to inertia which causes a displacement relative to the frame. This displacement is measured via capacitive, piezoelectric or piezoresistive sensors with the capacitive sensor the most applied. In capacitive accelerometers the proof



mass sits between two fixed plates and as the mass moves its distance to the plates changes resulting in a change in capacitance. These capacitance changes are converted in g-forces in three directions. MEMS accelerometers are small, light and have low power consumption and start-up times. Their main disadvantage is that they are not currently as accurate as accelerometers manufactured using traditional techniques. Linear accelerometers can be used as inertial navigation devices. Once an initial position is known, the accelerometer can track the change in position over time. Since the accelerometer used in this lab does not have a gyroscope, rotation cannot be measured directly[7]. The inertial navigation approach is sketched in Fig. 1.

Figure 1: Inertial navigation calculation methodology. Figure adapted from [8].

[7] This means that if you want to use this node to navigate, the orientation of the sensor should not change during the recording. With some trigonometry you should be able to find the angle from the 3D acceleration information.

[8] An introduction to inertial navigation

The position, *s* can be determined by:

$$v(t) = v(0) + \int_0^t (a(t) - g)dt \quad (2)$$

$$s(t) = s(0) + \int_0^t v(t)dt \quad (3)$$

with *a*(t) the temporal acceleration, *v*(t) the velocity, *s*(t) the position. *v*(0) and *s*(0) are the initial velocity and displacement, and g is the acceleration due to gravity that always acts on the sensor and thus the measurement needs to be corrected for the gravitation. 1 g = 9.81 m/s^2.

The integration scheme can be implemented on an MCU using the simple rectangular rule based on the rate with which the samples arrive.

$$v(t + \delta t) = v(t) + \delta t \cdot (a(t + \delta t) - g) \quad (4)$$

$$s(t + \delta t) = s(t0) + \delta t \cdot v(t + \delta t) \quad (5)$$

With δt the time between sample points.

To determine how the correction for the gravitational force needs to be applied look at the sketches of Fig. 2. The gravitational acceleration *g* will act against a weight going up and with a weight going down.

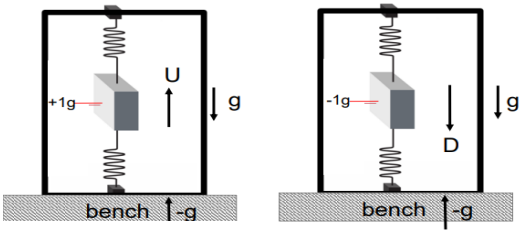


Figure 2: The sign of the correction by g when moving either up or down. Figure taken from [9].

A PPG sensor (Photoplethysmography sensor) measures blood volume changes in the microvascular bed of tissue using a light-based technology. It's commonly used in smartwatches, fitness trackers, and medical devices to monitor heart rate, and sometimes blood oxygen levels (SpO_2). The PPG chip used in this lab uses red (~660 nm) and IR (~940 nm) LED light. These wavelengths are chosen because they penetrate tissue well. The sensor works by light absorption or reflection. As blood pulses through the blood vessels with each heartbeat the small veins expands and contract leading to variations in light absorption and reflection. A photodiode or photodetector measures the amount of light that is either reflected (in reflective PPG) or transmitted (in transmissive PPG) through the tissue. In reflective PPG (implemented here) the photodiode and photodetector are at the same side of the chip. The photodetector output over time will give a voltage waveform that varies over time with a pattern related to heart rate. Fig. 3 gives an illustration of the traditional position of a PPG sensor and the theoretical signal waveform that is produced.

SpO_2 (peripheral capillary oxygen saturation) can be derived from red and IR light reflection because oxygenated and deoxygenated haemoglobin absorb light differently at specific wavelengths. Oxyhaemoglobin (HbO_2) absorbs more IR light and less red light while deoxyhaemoglobin (Hb) absorbs more red light and less IR light. The ratio of absorption at these two wavelengths changes depending on the oxygen saturation of the blood. SpO2 can be calculated by:

$$R = \frac{(AC_{red}/DC_{red})}{(AC_{IR}/DC_{IR})} \quad (6)$$

with *AC* the pulsatile component (due to arterial blood) and *DC* the non-pulsatile component (due to skin, tissue, venous blood). This ratio *R* must then be mapped to an SpO2 value using a calibration curve derived from empirical data. SpO2 levels should be > 95% for healthy people.

Fig. 3(b) shows that the signal has two phases related to the mechanism behind the operation of the heart driving blood through the veins. The heart rate can be extracted from the distance between the peaks in the waveform.

[9] Slide 1

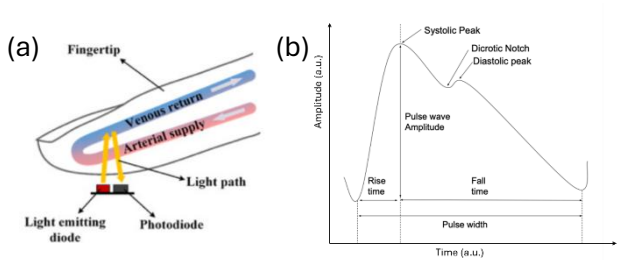


Figure 3: (a) Schematic position on the index finger of the LED and photodetector. (b) The part of the waveform that is repeated as a function of time. The signal in this figure does not have noise nor baseline wander.

Note that real sensors will have noise related to the electronic components as well as motion-related noise. For the PPG sensor, the signal will vary with any variation of the pressure between the sensor and the skin. In addition, low frequency motion noise that is a result of the sensor moving with respect to the skin surface with generate unwanted low frequency noise. Don't forget that mains noise (50 Hz in the UK) and other noise sources (lights, electronic switches, etc) will distort the signal.

Therefore, the signals from both sensors will require filtering (bandpass) to remove different unwanted noise components.

Appendix B: A comparison between the I^2C and SPI.

Feature	I^2C	SPI
Wires	2 (SDA, SCL)	4 (MOSI, MISO, SCLK, SS)
Complexity	Simpler wiring	More pins needed, especially with multiple devices
Addressing	Devices have unique addresses	Each device needs a separate chip select (SS) line
Typical Speed	Up to 400 kHz (standard), 3.4 MHz (high-speed)	Up to tens of MHz
Performance	Slower	Faster, better for high-speed data transfer
Communication	Half-duplex	Full-duplex
Master/Slave	Multi-master possible	Single master, multiple slaves
Protocol	More complex (start/stop conditions, ACK/NACK)	Simpler, just clock and data lines
Overhead	Higher	Lower

Summary

- Use I^2C when you want simplicity and fewer pins, and speed isn't critical. Good for low-speed peripherals like temperature sensors, RTCs, EEPROMs.
- Use SPI when you need speed and performance and can afford more pins. Good for high-speed devices like displays, ADCs, and flash memory.

I^2C PCB implementation guidelines:

- Use a 4-layer PCB for improved grounding, signal – ground separation and easier routing.
- Proper impedance matching must be done.
- The total bus capacitance should be kept < 400 pF, thus keep traces short, direct, avoid large pads, stubs and branches and minimize via count. Run the SDA and SCL line not too close together.

Kristel & Prabhav & Sara – 06/10/2025

- Use proper pull-up resistors: 2.2 kΩ to 10 kΩ[10], place them close to the master (MCU).
- Place a solid ground plane under the I^2C traces and use guard vias to ground for shielding.
- Add test points on SDA and SCL lines for oscilloscope probing and debugging.
- For more detailed guidance, you can refer to:
  - NXP I^2C-bus Specification and User Manual (UM10204)
  - Analog Devices – Designing Robust, Isolated I^2C Interfaces

[10] What are I2C Pull-Up Resistors and How to Calculate their Values - Total Phase