POLITECNICO
MILANO 1863

Final Projects

Instructions

Each student can choose one final project in groups of either 3 or 4 students. All projects require the same skillset, acquired during the course, and the same effort-per-person. You will have to **meet all the project requirements** to complete the project.

Project Meetings

The kick-off meeting of the projects will take place after their presentation, on April 20. Project tutors will be available in the following days for helping you in carrying out the final project activities:

- Wednesday, April 27
- Wednesday, May 4
- Wednesday, May 11
- Wednesday, May 18
- Wednesday, May 25

Each group should send an e-mail to the project tutor by the day before the meeting with questions and doubts to be solved so that a proper timeframe can be allocated. There will be a mid-term presentation at the end of the semester (most probably on Wednesday, June 1; check on WeBeep for confirmation).

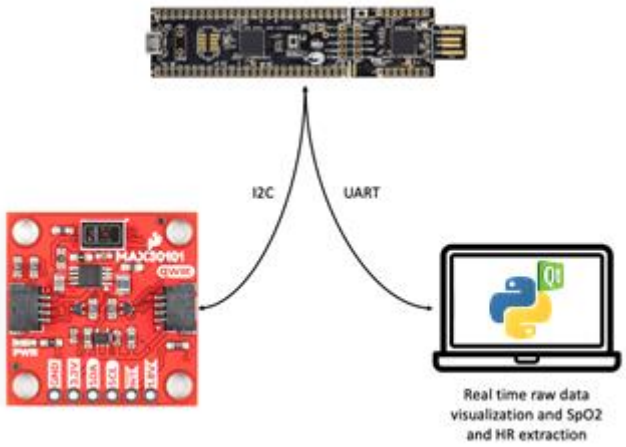
Project Delivery

The project must be submitted using **GitHub**. The final version, which will be evaluated by the professor and the tutors, will be the one corresponding to the last commit before the project deadline. You will be added as a collaborator to a project repository owned by the ltebs-polimi account and you will have to use that repository to version your project.

POLITECNICO
MILANO 1863

Project 1

Tutor: Andrea Rescalli



In this project, students are required to design and develop a system able to perform oxygen saturation and heart rate measurements using an external sensor (photodetector) mounted on a breakout board, and to visualize the collected data on a PyQt5-based GUI. You can power your board using the USB connector on the KitProg, but for those of you who want to move around (and are, of course, not bounded to the USB cable for UART communication), it is possible to power the board using the micro-usb port on the PSoC CY8CKIT-059 target side.

The photodetector board must be powered only from a 3.3 voltage source, and the SDA and SCL lines of the photodetector board are not 5V-tolerant. **Therefore, you must use a voltage regulator and a level-shifter to power and communicate with the photodetector board!** If you're not sure on the wiring, please contact your tutor.

The main objectives of the project are:

- Acquire raw data from the photodetector breakout board using its built-in FIFO to reduce the load of the CPU.
- Process the raw data in order to extract oxygen saturation or heart rate value.
- Provide the end-user with a GUI to visualize the collected raw data and the processed oxygen saturation and heart rate values.
- Develop a custom case and a PCB to minimize interferences during the measurements.

Project Requirements

General requirements:

1. Students should design a 3D printed case for the MAX30101 breakout board in order to avoid the interference of lights on the photodiodes of the breakout board.
2. The MAX3010 must be configured to temporarily store data in the internal FIFO to reduce the CPU load on the PSoC MCU.

3. As the product is designed to be wearable, the system should be designed in such a way that power consumption is reduced. You can exploit PSoC sleep modes, and power off the MAX30101 when needed.

Specific requirements: the students are required to perform a measurement of heart rate, oxygen saturation (SpO2), and to visualize both the raw data from the light channels and the processed data (HR and SpO2) on a PyQt5-based GUI, from which the system must be also configurable.

1. At start-up time, the MAX30101 must be configured to work properly for SpO2 and HR measurement.
2. The functions used to interact with the MAX30101 must be properly organized in header and source files.
3. A PyQt5-based GUI must be developed to configure the settings for SpO2 and HR measurements, and to retrieve both raw data and processed data (SpO2 and HR). These are the basic functionalities required for the GUI to operate:
    a. Automatic detection of the COM port: this can be performed by sending a custom command (e.g. **v**) from the host side, to which the PSoC must answer in a determinate way (e.g. **Heart Rate $$$**). Upon detection of the expected answer, then the COM port is open and connection occurs.
    b. The GUI should allow to start/stop data streaming from the device side. A command of **b** should start data streaming, a command of **s** should stop data streaming.
    c. The following options should be configurable from the GUI side. The GUI should also be able to retrieve the current settings stored on the device and show them to the user. The settings should be stored in the internal EEPROM memory so that they are properly retrieved at start-up time:
        i. SpO2 Sample Rate control: see values in Table 6 of the device datasheet.
        ii. LED Pulse Width control: see values in Table 7 of the device datasheet.
        iii. SpO2 ADC Range control: see values in Table 5 of the device datasheet.
        iv. LED Current Control: see values in Table 8 of the device datasheet.
    d. If you think that additional configuration should be added, please feel free to do so.
    e. You may decide to implement the code for SpO2 and HR computation in the GUI or on the PSoC side: this is completely up to you.
    f. The GUI should visualize the raw channels data in graphical form and also the computed SpO2 and HR values
4. If the MAX30101 is not found on the I2C bus either at start-up or at runtime, then this must be shown to the user in the GUI (through a simple label) and through the on-board LED:
    g. If the LED is always ON, then everything is properly connected and working
    h. If the LED is ON/OFF with a 50% duty cycle and a period of 1 sec, then the MAX30101 is not connected. In this condition, the GUI must disable the option to start data streaming

Material List

- Cypress CY8CKIT-059 PSoC 5LP Prototyping Kit
- MAX30101 Breakout Board by SparkFun: https://www.sparkfun.com/products/16474
- Logic Level Shifter by SparkFun: https://www.sparkfun.com/products/12009
- 3.3 Voltage Regulator

Hardware Setup

Please, use the following hardware setup to connect all the devices required for the project to the PSoC.

Component	PIN	Port (PSoC)	Project
MAX3010	SCL	12.0	All
MAX3010	SDA	12.1	All
MAX3010	INT	12.4	All
LED		2.1	All
UART_RX		12.6	All
UART_TX		12.7	All

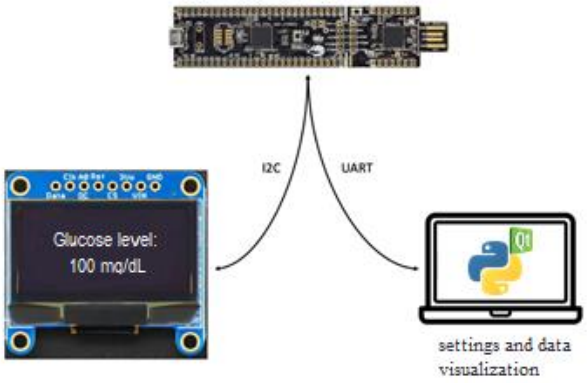
Project References

1. How to Design Peripheral Oxygen Saturation (SpO2) and Optical Heart Rate Monitoring (OHRM) Systems Using them AFE4403: https://www.ti.com/lit/an/slaa655/slaa655.pdf
2. Pulse Oximeter Fundamentals and Design : https://www.nxp.com/docs/en/application-note/AN4327.pdf
3. Penetration Depth Guide for Biosensors Applications: https://www.maximintegrated.com/en/design/technical-documents/app-notes/6/6433.html
4. Datasheet MAX30101: https://cdn.sparkfun.com/assets/8/1/c/9/0/MAX30101_Datasheet.pdf
5. MAX30101 Breakout Board hookup guide: https://learn.sparkfun.com/tutorials/sparkfun-photodetector-max30101-hookup-guide/hardware-overview
6. SparkFun MAX30101 Arduino library (this library is to be used as a starting point, it has to be adapted to your needs): https://github.com/sparkfun/SparkFun_MAX3010x_Sensor_Library
7. SparkFun Level Shifter Breakout Board: https://www.sparkfun.com/products/12009
8. Level Shifter Hookup Guide: https://media.digikey.com/pdf/Data%20Sheets/Sparkfun%20PDFs/Bi-Directional-Logic-Level_HookupGuide.pdf
9. 3.3 V Voltage Regulator datasheet: https://www.ti.com/general/docs/suppproductinfo.tsp?distId=10&gotoUrl=http%3A%2F%2Fwww.ti.com%2Flit%2Fgpn%2Fua78m

POLITECNICO
MILANO 1863

Project 2

Tutor: Andrea Rescalli



In this project, students are required to design and develop a system able to perform glucose monitoring using a microcontroller to be setup as a potentiostat together with commercially available glucose strips. Data visualization happens in two steps: a simple display will inform the final user on the main result of the measurement (i.e. the glucose concentration); in parallel, a PyQt-based GUI has to be developed taking into consideration a 'clinical' use: an operator can connect the device to a host machine and can both configure the parameters of the measurement and visualize the raw data acquired.

You can power your board using the USB connector on the KitProg, but for those of you who want to move around (and are, of course, not bounded to the USB cable for UART communication), it is possible to power the board using the micro-usb port on the PSoC CY8CKIT-059 target side.

The main objectives of the project are:

- Implement a PSoC-based potentiostat that can perform basic electrochemical measurements such as cyclic voltammetry and chronoamperometry.
- Application-oriented design of the device with custom case and PCB.
- Process the raw data in order to extract glucose levels: a calibration of the device will be needed.
- Implement two scenarios for the visualization of data: end user and clinician.

Project Requirements

General requirements:

1. Students should design a 3D printed case that includes the microcontroller and has to interface with the glucose strips: be careful, the pads disposition of the glucose strips you select must be of easy access so that you can couple the sensor with your device in a simple manner. A PCB (to be designed and realized) is mandatory to perform the coupling. You can experiment as much as you want.
2. The PSoC must be able to perform both cyclic voltammetry and chronoamperometry measurements, and the clinician must be able to change the parameters of the measurements as well as easily visualize and access the data.

3. As the product is designed to be wearable, the system should be designed in such a way that power consumption is reduced. You can exploit PSoC sleep modes, power off the OLED and unnecessary components when not needed and so on.

Specific requirements: the students are required to perform a measurement of glucose concentration exploiting commercially available strips (to be chosen by the students according to their design) and to display these values on the external display. Furthermore, a GUI should be developed to simulate a system configuration by a clinician and to visualize data. Here is the detailed list of the requirements:

1. The external OLED 128x64 display must be configured to communicate using the I2C bus. Remember that the I2C bus requires pull-up resistors. Check on the description of the breakout boards of the display (link provided at the end of the project description) to check whether pull-ups resistors were already integrated in the breakout board.
2. The actual measurement result must be shown on the display. You can decide what to show at start-up time (a logo, a string,..). You can also decide to set up visual animations during the measurement, it's completely up to you. When the device enters a "sleep" mode, all the display pixels should be turned off.
3. A reference paper to guide you in the configuration of the PSoC as a potentiostat is provided in the references section. The paper is to be used as a guide, modify the design in order to implement your own version compatible to your needs. All the firmware files must be properly organized in header and source files.
4. A PyQt5-based GUI must be developed to configure the settings for cyclic voltammetry and chronoamperometry measurements (both for the current session and as default values) and to retrieve both raw data (current vs voltage for cyclic voltammetry and current vs time for chronoamperometry) and processed data (the glucose concentration level). These are the basic functionalities required for the GUI to operate (additional info can be found at points 5 to 8):
    a. Automatic detection of the COM port: this can be performed by sending a custom command (e.g. **v**) from the host side, to which the PSoC must answer in a determinate way (e.g. **Glucose $$$**). Upon detection of the expected answer, then the COM port is open and connection occurs.
    b. The GUI should allow to start/stop a measurement. A command of **b** should start data streaming, a command of **s** should stop data streaming.
    c. The following options should be configurable from the GUI side. The GUI should also be able to retrieve the current settings stored on the device and show them to the clinician. The settings should be stored in the internal EEPROM memory so that they can be adopted for user measurements:
        i. Cyclic voltammetry scan rate.
        ii. Cyclic voltammetry initial/final potential values.
        iii. Chronoamperometry working potential.
        iv. Chronoamperometry acquisition time.
    d. If you think that additional configuration should be added, please feel free to do so.
    e. The GUI should visualize the raw data in graphical form and also the computed glucose concentration values
5. The clinician should be able to modify

   a. For cyclic voltammetry: **scan rate** and **initial/final potential values** for the scan (tip: create a function that generates a parametric triangular wave according to these values, which will be parameters to be passed in input to the function).
   b. For chronoamperometry: the **fixed voltage** at which the working electrode is kept during the measurement and the **time** after which the measurement is stopped.
6. The user will automatically perform a measurement based on default values chosen appropriately by the manufacturer of the device (= you) after a calibration: this step is fundamental in order to retrieve
   a. The optimal voltage at which chronoamperometry will be performed (is the voltage, in the cyclic voltammetry measurement, at which a peak current is observed).
   b. The calibration curve, that allows you to obtain the mathematical relationship to links the measured current to the glucose concentration in the sample under analysis.

   N.B: *the user has only the option to start the chronoamperometric measurement (e.g. by pressing a button), it will be performed according to stored values.*

   N.B.B: *initially you will test the architecture on reference resistors, then for the calibration and performance you won't be asked to prick your finger. You can look in literature (even in the reference paper) for experiments that can be carried out to perform this measurement. We can also create solutions of known glucose concentration*.

7. Values obtained during the calibration must be stored in the internal EEPROM of the microcontroller so that they can be accessed at any time. The clinician must have the option, from the GUI, to visualize and eventually modify these default values according to his needs.
8. The following packet structure is used to configure all the device settings: `[header, data, tail]`. The header is always equal to 0xA0, while tail vary in order to distinguish between the various paramaters (i.e. scan rate, initial/final values, fixed voltage, time)
   a. If you think that additional configuration should be added, please feel free to do so by following the same protocol described above
   b. The settings should be stored in the internal EEPROM memory so that they are properly retrieved at start-up time

Material List

- Cypress CY8CKIT-059 PSoC 5LP Prototyping Kit
- Display OLED 128x64 Monochrome by Adafruit: https://www.adafruit.com/product/938
- Commercial glucose strips: to be chosen by students according to their design. Select these as soon as possible for two reasons: they influence the development of your device and we need to buy them.

Hardware Setup

Please, use the following hardware setup to connect all the devices required for the project to the PSoC.

Component	PIN	Port (PSoC)
UART_RX		12.6
UART_TX		12.7
OLED DISPLAY	SCL	12.0
OLED DISPLAY	SDA	12.1

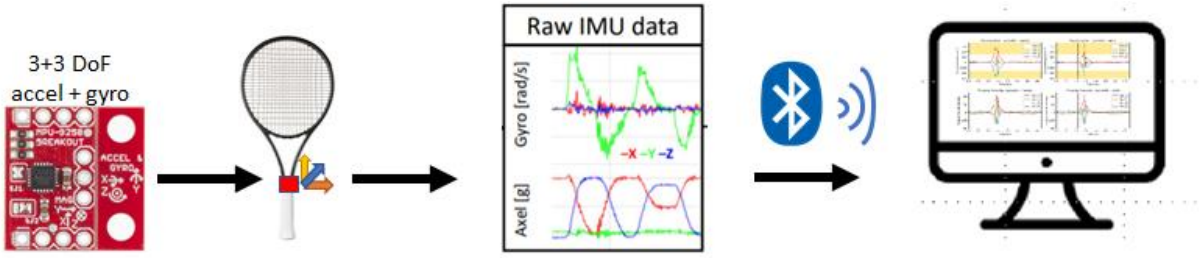
Project References

1. PSoC as potentiostat: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0201353#sec002
2. Beginner guide to CV: https://pubs.acs.org/doi/10.1021/acs.jchemed.7b00361
3. Adafruit SSD1306 C++ Library: https://github.com/adafruit/Adafruit_SSD1306
4. SSD1306 Datasheet: https://cdn-shop.adafruit.com/datasheets/SSD1306.pdf
5. SSD1306 Useful Information: https://iotexpert.com/debugging-ssd1306-display-problems/
6. SSD1306 PSoC Library (this library is to be used as a starting point, you should provide improvements to it): https://github.com/derkst/Cypress-PSOC-OLED

POLITECNICO
MILANO 1863

Project 3

Tutor: Matteo Rossi



In this project, the students are required to design a smart tennis racket able to monitor and classify technical tennis movements exploiting the PSoC and the 3-axis accelerometer LIS3DH. The device will have to implement an algorithm for gesture recognition and data visualization on a GUI. Communication between PSoC and PC will be made with HC05 Bluetooth module, in order to make the device pluggable on a real racket. The PSoC will communicate with the LIS3DH over I2C (Fast Mode, 400 kHz) to acquire data at the best full-scale range and sampling frequency to meet the requirements listed below. You must power your board using a battery.

The main objectives of the project are:

- Acquire raw data from the 3-axis accelerometer using its built-in FIFO to reduce the load of the CPU.
- Process the raw data.
- Provide the end user with a GUI to visualize the collected raw data and the processed data.
- Students should design a 3D printed case that includes the microcontroller and the whole instrumentation.
- Print your own PCB to reduce instrumentation bulkiness.

Project Requirements

1. Design of PCB
2. Design of 3D printed chassis
3. The functions used to interact with the LIS3DH must be properly organized in header and source files.
4. A GUI must be developed to display and classify either raw and postprocessed data. These are the basic functionalities required for the GUI to operate:
   - Automatic detection of the COM port: this can be performed by sending a custom command (e.g., **v**) from the host side, to which the PSoC must answer in a determinate way (e.g., **Tennis$$$**). Upon detection of the expected answer, then the COM port is open, and connection occurs.
   - The GUI should allow to start/stop data streaming from the device side. A command (e.g., '**b**') should start data streaming, a command (e.g., '**s**') should stop data streaming.
   - The following options should be configurable from the GUI side. The GUI should also be able to retrieve the current settings stored on the device and show them to the user. The settings should be stored in the internal EEPROM memory so that they are properly retrieved at start-up time:
     a. LIS3DH Measurement range (FS): see values in Table 4 of the device datasheet.

   b. LIS3DH Sensitivity (So): see values in Table 4 of the device datasheet.
   c. If you think an additional configuration should be added, please feel free to do so.
5. If the LIS3DH is not found on the I2C bus either at start-up or at runtime, then this must be shown to the user in the GUI (through a simple label) and through an LED:
   a. If the LED is always ON, then everything is properly connected and working.
   b. If the LED is ON/OFF with a 50% duty cycle and a period of 1 sec, it means the device is not connected. In this condition, the GUI must disable the option to start data streaming.

Material List

- PSoC CY8CKIT-059
- LIS3DH 3-axis accelerometer
- HC-05 bluetooth serial module

Hardware Setup

The following is a suggestion for the hardware setup to connect all the devices required for the project to the PSoC.

Component	PIN	Port (PSoC)
LIS3DH	SCL	12.0
LIS3DH	SDA	12.1
LIS3DH	INT	12.4
HC-05	RX	2.3
HC-05	TX	2.4
LED	-	2.1

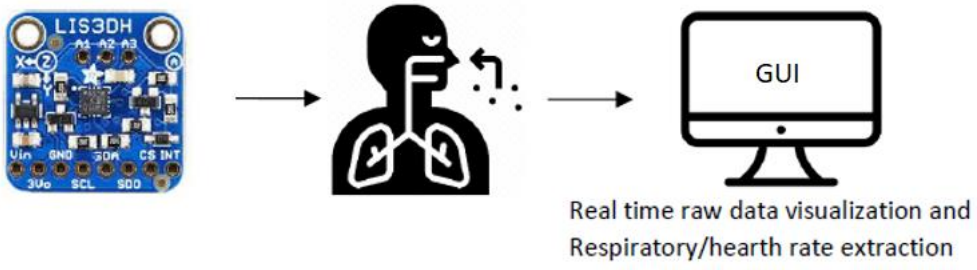
Project References

- HC-05 Datasheet: Link
- Adafruit LIS3DH C++ Library (for Arduino): Link
- LIS3DH Datasheet: Link
- Literature Overview: Link

POLITECNICO
MILANO 1863

Project 4

Tutor: Luca Marsilio



In this project, the students are required to develop a simple **Respiratory / Heart Rate Monitor** exploiting the PSoC and the 3-axis accelerometer LIS3DH. The device will have to implement an algorithm for respiratory and heart rate computation and data visualization on a GUI. Communication between PSoC and PC will be made with HC05 Bluetooth module, in order to make the device wearable. The PSoC will communicate with the LIS3DH over I2C (Fast Mode, 400 kHz) to acquire data at the best full-scale range and sampling frequency to meet the requirements listed below. You can power your board using the USB connector on the KitProg, but for those of you who want to move, it is possible to power the board using the micro-USB port on the PSoC CY8CKIT-059 target side with e.g., a power bank.

The main objectives of the project are:

- Acquire raw data from the 3-axis accelerometer using its built-in FIFO to reduce the load of the CPU.
- Process the raw data to obtain the respiratory and heart rate values.
- Provide the end user with a GUI to visualize the collected raw data and the processed respiratory/heart rate values.
- Students should design a 3D printed case that includes the microcontroller and the whole instrumentation.
- Students should design their own PCB to reduce instrumentation bulkiness.

Project Requirements

In this project, the students are required to perform a measurement of respiratory rate (RR), heart rate (HR) and to visualize both the raw data from the 3-axis accelerometer LIS3DH and the extracted measure on a GUI, from which the system must be also configurable.

1. At start-up time, the system must retrieve the latest configuration stored in the internal EEPROM.
2. The functions used to interact with the LIS3DH must be properly organized in header and source files.
3. A GUI must be developed to configure the settings for either RR or HR measurements, to retrieve the corresponding raw data and get the current value. These are the basic functionalities required for the GUI to operate:
    - Automatic detection of the COM port: this can be performed by sending a custom command (e.g., **v**) from the host side, to which the PSoC must answer in a determinate way (e.g., **Breath $$$**). Upon detection of the expected answer, then the COM port is open, and connection occurs.
    - The GUI should allow the selection of either the respiration rate or heart rate measurement.

- The GUI should allow to start/stop data streaming from the device side. A command (e.g., '**b**') should start data streaming, a command (e.g., '**s**') should stop data streaming.
- The following options should be configurable from the GUI side. The GUI should also be able to retrieve the current settings stored on the device and show them to the user. The settings should be stored in the internal EEPROM memory so that they are properly retrieved at start-up time:
    i. LIS3DH Measurement range (FS): see values in Table 4 of the device datasheet.
    ii. LIS3DH Sensitivity (So): see values in Table 4 of the device datasheet.
    iii. If you think an additional configuration should be added, please feel free to do so.

You may decide to implement the code for RR/HR computation in the GUI or on the PSoC side: this is completely up to you.

- The GUI should visualize the raw channels data and the computed RR/HR values.

If the LIS3DH is not found on the I2C bus either at start-up or at runtime, then this must be shown to the user in the GUI (through a simple label) and through the on-board LED:

If the LED is always ON, then everything is properly connected and working.

- If the LED is ON/OFF with a 50% duty cycle and a period of 1 sec, it means the device is not connected. In this condition, the GUI must disable the option to start data streaming.

Material List

- PSoC CY8CKIT-059
- LIS3DH 3-axis accelerometer
- HC-05 bluetooth serial module

Hardware Setup

The following is a suggestion for the hardware setup to connect all the devices required for the project to the PSoC.

Component	PIN	Port (PSoC)
LIS3DH	SCL	12.0
LIS3DH	SDA	12.1
LIS3DH	INT	12.4
HC-05	RX	2.3
HC-05	TX	2.4
UART-RX	RX	12.6
UART-TX	TX	12.7
LED (Internal)		2.1

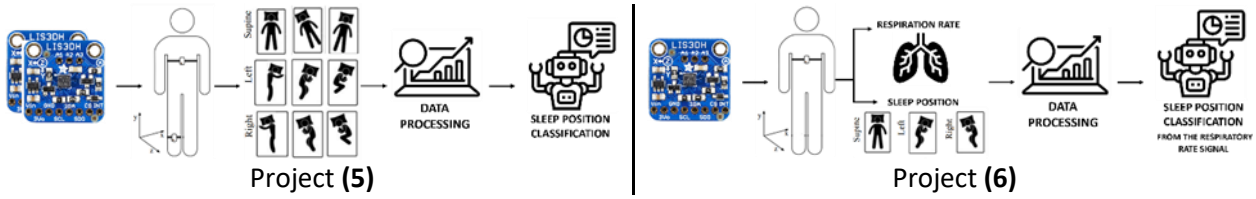
Project References

- HC-05 Datasheet: Link
- SSD1306 Datasheet: Link
- Adafruit LIS3DH C++ Library (for Arduino): Link
- LIS3DH Datasheet: Link
- Literature Overview: Link

POLITECNICO
MILANO 1863

Project 5 and 6

Tutor: Giulia Alessandrelli



Project **(5)** Project **(6)**

In this project, the students are required to develop a system able to correctly classify typical sleep positions, using an accelerometer-based wearable device. The project includes the development of the wearable device, data acquisition, data processing, and the development of a system able to perform sleep position classification.

The project exploits the PSoC and the 3-axis accelerometer LIS3DH. Communication between PSoC and PC will be made with HC05 Bluetooth module, to make the device wearable. The PSoC will communicate with the LIS3DH over I2C (Fast Mode, 400 kHz) to acquire data at the best full-scale range and sampling frequency to meet the requirements listed below. You can power your board using the USB connector on the KitProg, but it is possible to power the board using the micro-USB port on the PSoC CY8CKIT-059 target side with e.g. a power bank.

Project **(5)** asks to perform sleep position classification in a high-detailed way by combining information from two accelerometers positioned on the chest and on one ankle. On the other hand, project **(6)** asks to perform sleep position classification by extracting information from the respiration rate.

Once the wearable system is ready, the students have to perform data acquisition according to a specific protocol that simulates the typical sleep positions: this allows to have labeled data, thus it increases the number of methodologies available to perform the classification.

The students are highly encouraged to explore multiple classification methods (both unsupervised and supervised). The final goal is to develop a model, built on the acquired data, able to classify unseen data correctly. Indeed, a smaller portion of the total acquired data has to be left out from the training phase of the model, to have the so-called “test set” available to quantify the real model performance.

The **main objectives** of the project are:

- Application-orientated design of the wearable device with custom case (optional) and PCB (mandatory).
- Raw data acquisition from the 3-axis accelerometer using its built-in FIFO to reduce the load of the CPU.
- Project **(5)**. Process the raw data and correctly label the data.

Project **(6)**. Process the raw data, label the data, and compute the respiratory rate signal.

- Compute multiple classification methods (mandatory) and eventually provide a model, built from the collected raw data, able to correctly perform sleep position classification.

- The goal of project **(5)** is to be able to forecast sleep position, starting from the position data acquired by both the accelerometers.
- The goal of project **(6)** is to be able to forecast sleep position, starting from the respiration signal.

General Requirements:

- The functions used to interact with the LIS3DH must be properly organized in header and source files.
- The code(s) regarding data processing and classification must be properly organized. It is suggested the use of Python (in Google Colaboratory or Jupyter Notebook), but we're open to other coding languages too.
- Documentation on how you set up the hardware (1 page) needs to be provided by the students.
- Collaboration between teams is highly encouraged.

Specific Requirements:

- The project **(5)** uses two accelerometers. Students have to be positioned one on the chest and the second one on the ankle. It is mandatory to integrate the signals acquired from both the accelerometers.
- The project **(6)** uses an accelerometer, that has to be positioned on the chest.
- Data from both the accelerometers has to be integrated and preprocessed, filtering out the potential noise.
- Data acquisition has to be performed according to the agreed protocol.
- 10 subjects (minimum) have to be involved, to gain a sufficient inter-subject variability. The suggestion is to perform data acquisition According to the classification method, it is mandatory to split the acquired data in training and test set (70/30).
- Exploring the scientific literature regarding methodology about sleep position classification is mandatory.

Data Acquisition Protocol:

- Project **(5)**
  - 12 sleep positions have to be tested. As a reference, see the appendix.
  - Simultaneous data acquisition from both the accelerometers has to be done.
  - Each subject has to keep the position for 90 seconds. The protocol duration last 18 minutes.
- Project **(6)**
  - 4 sleep positions have to be tested: supine, prone, lateral left, and lateral right. As a reference, see the appendix.
  - The signal related to the position and the one related to the respiratory rate have to be computed and saved separately. Pay attention to keep the information about which signal belong to which subjects.
  - Each subject has to keep the position for 3 minutes. The protocol duration last 12 minutes.

Adjustments and modification in the suggested protocol are possible, from both sides, after discussion.

Materials List

- PSoC CY8CKIT-059 (1 or 2 according to the hardware setup chosen by the students)
- LIS3DH 3-axis accelerometer(s)
- HC-05 bluetooth serial module

If the students believe that any other material could be beneficial for their project, we're open to discuss it.

Hardware Setup

The following table is an example of hardware setup. However, it's up to you to decide how to connect all the devices required for the projects.

Component	PIN	Port (PSoC)
LIS3DH	SCL	12.0
LIS3DH	SDA	12.1
LIS3DH	INT	12.4
HC-05	RX	2.3
HC-05	TX	2.4
UART_RX*	RX	12.6
UART_TX*	TX	12.7
LED (internal)		2.1

Project References

- HC-05 Datasheet: Link
- SSD1306 Datasheet: Link
- Adafruit LIS3DH C++ Library (for Arduino): Link
- LIS3DH Datasheet: Link

Literature

- Application of Acceleration Sensors in Physiological Experiments. (2014) 10.2478/jee-2014-0049
- Wearable Accelerometer Based Extended Sleep Position Recognition. (2017) 10.1109/HealthCom.2017.8210806
- Sleep classification from wrist-worn accelerometer data using random forests. (2021) 10.1038/s41598-020-79217-x

Appendix

Project **(5)**	Project **(6)**