TÉCNICO LISBOA

Technologies of Computing Systems

1st Project

Pedometer using IMU, Ultrasonic Sensor and BLE

1st Semester, Second Quarter 2025/26

Leonel Sousa, Diogo Caetano, Ruben Afonso

November 2025

1 Introduction

This project involves the design and implementation of a pedometer system using an Arduino [1] Nano 33 BLE platform equipped with an nRF52840 Microcontroller bearing an Arm Cortex-M4 32-bit Processor [2], and 12-bit ADCs. Additionally, the device integrates data from the on-board inertial measurement unit (IMU), an external ultrasonic sensor, and can transmits the processed information wirelessly through Bluetooth Low Energy (BLE). The goal of this project is to detect and count steps using IMU data, while employing ultrasonic measurements to provide additional contextual information, such as estimating the distance to a nearby wall to calculate average step length and detecting situations where the device is moving without actual steps. No example code is provided; instead, technical objectives and constraints are specified to simulate a real client specification scenario. Additional information about the specific Arduino platform to be used in this project (i.e. Nano 33 BLE) can be found on the products page [3].

2 Material and Devices

The following components are required:

i. Breadboard
ii. Arduino Nano 33 BLE — https://docs.arduino.cc/hardware/nano-33-ble
iii. USB Cable
iv. Ultrasonic Sensor – HC-SR04 — [https://www.electronicwings.com](https://www.electronicwings.com/sensors-modules/hc-sr04-ultrasonic-sensor)
v. Computer for programming and BLE monitoring
vi. **Optional (Bonus):**
   • Vishay IR LED (emitter) + series resistor (per datasheet)
   • Vishay IR phototransistor (receiver) + bias resistor (or transimpedance stage)

3 Goals

This project targets the following main objectives:

1. Implement low-level I^{2}C communication to access acceleration data from the LSM9DS1 IMU.
2. Implement ultrasonic distance measurements through digital pins and timers, without using high-level functions such as `pulseIn()`.

3. Develop a step counting algorithm using acceleration data (e.g., peak detection methods).
4. Use ultrasonic measurements to estimate the distance to a nearby wall and, by combining this with step counts, calculate the average step length during walking.
5. Use ultrasonic measurements to detect situations where the device is moving without actual steps (e.g., shaking).
6. Transmit step count, step length, and relevant sensor data through BLE to another device.
7. Ensure that data acquisition and processing run continuously and efficiently using timers and interrupts.

4 Sensors

4.1 Inertial Module

The LSM9DS1 module in the Arduino platform is a 9-axis inertial sensor [4] . It can sense 3 acceleration channels, 3 angular rate channels and 3 magnetic channels. To access the sensor tilt, required in this project, only some of those components are necessary. However, more complex implementations are considered also for the final grade.

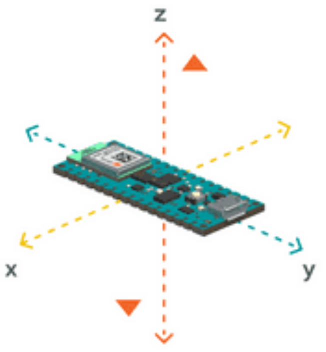

Figure 1: Arduino Nano 33 BLE accelerometer coordinate system.

4.2 Ultrasonic sensor

The HC-SR04 [5] employs ultrasonic echo-based distance estimation. The proximity measurement starts when a pulse of at least 10 µs is applied to the Trigger pin of the module. The sensor will emit a burst of sonic pulses with a known signature that allows the system to differentiate from interfering signals. The pulses travel through the air (Figure 2). During this time the Echo pin goes High. If there is no reflection the Echo pin will timeout after 38ms. If the emitted pulses are reflected, the echo pin goes low as soon as the signal is received. This produces a pulse duration related to the distance of the reflecting obstacle, which can be used to compute the said distance.

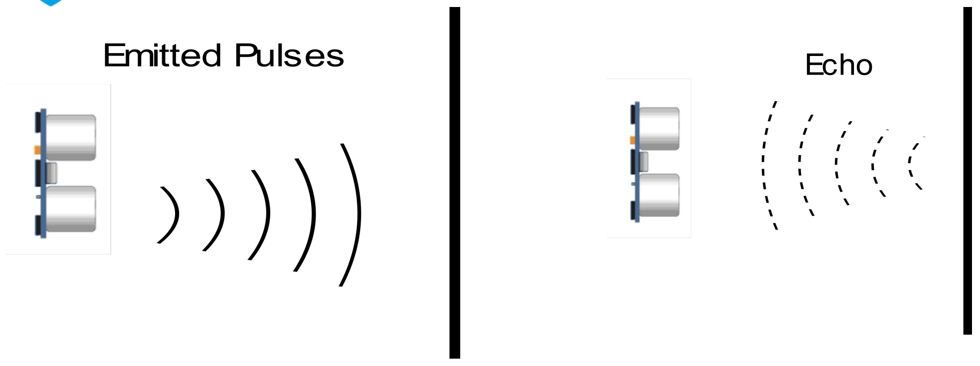

Figure 2: Ultrasonic signal emission and subsequent eco.

4.3 Infrared sensor

The infrared sensor uses infrared light reflection as a means of proximity estimation. This sensor is based on an emitter LED [6] and a receiving photodiode [7]. As with the ultrasonic sensor, the emitted light is only reflected in the presence of an obstacle. However, in the case of light, computing the delay between emitted and reflected photons would be demanding. Thus, in this case the distance is correlated with the amount of light that is reflected, and consequently excites the photodiode (Figure 3).

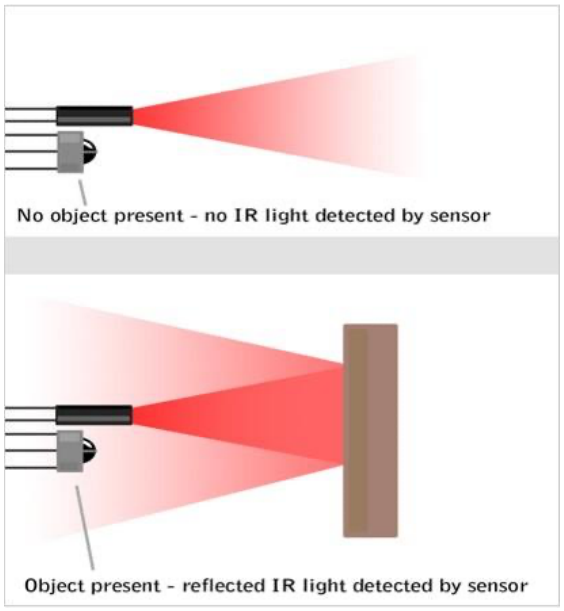
No object present - no IR light detected by sensor
Object present - reflected IR light detected by sensor

Figure 3: Infrared sensor refraction schematic from.

Relevant keywords for research: 'HC-SR04 timing diagram', 'ultrasonic echo measurement Arduino timers', 'attachInterrupt echo pin', 'non-blocking ultrasonic distance measurement'.

5 BLE Communication

Bluetooth Low Energy (BLE) is used to transmit step counts and additional sensor data to an external device (e.g., a computer or smartphone). The Arduino Nano 33 BLE has built-in BLE support, and the use of BLE libraries is allowed in this project. Students must configure BLE advertising and characteristics to transmit relevant data periodically. **A third-party BLE client application may be used to visualize the data** (e.g., smartphone BLE scanner apps or desktop BLE tools) without requiring a custom app.

Reference:https://docs.arduino.cc/tutorials/nano-33-ble-sense/ble-device-to-device

6 Step Detection Algorithm

Step detection is based on analyzing vertical acceleration signals to identify periodic patterns corresponding to steps. A simple but effective method involves applying filtering to remove high-frequency noise and then detecting peaks that exceed a defined threshold. Adaptive thresholding techniques can improve robustness across users and walking styles.

Relevant keywords for research: 'peak detection in acceleration signals', 'moving average filter for pedometer', 'adaptive threshold step counting', 'IMU vertical axis orientation'.

7 Implementation Details

• The system must ensure that IMU data is acquired at a consistent sampling rate (e.g., 50–100 Hz), either by configuring the sensor's internal output data rate or by implementing a suitable scheduling mechanism in the firmware (e.g., using timers).

• External interrupts should be used for ultrasonic echo pulse measurement.

• The internal mechanisms of timers and I^2C must be understood and documented (e.g., prescaler, counting cycles).

• Avoid using functions such as pulseIn(), noTone(), tone(), shiftIn(), shiftOut(), and other high-level abstractions.

• Ensure BLE communication does not block acquisition routines.

7.1 Electrical connections

The electronic circuitry developed must be implemented in a single breadboard powered exclusively by Arduino. The Arduino module must be placed in the breadboard to allow connections with the peripherals, as depicted in Figure 4. There are two voltages available at the Arduino as long as it plugs via USB; +5 V voltage is <u>available</u> on the VIN pin (this pin is named VIN because it also accepts external power in case USB is not connected) and +3.3 V is available on the +3V3 pin. Note that +5V pin is not enabled by default.

CAUTION: The Arduino module core operates with a supply voltage of 3.3V. All input/output pins only support 3.3V. The ultrasound module needs +5V VCC so it is necessary to make a level shift from the ECHO output.

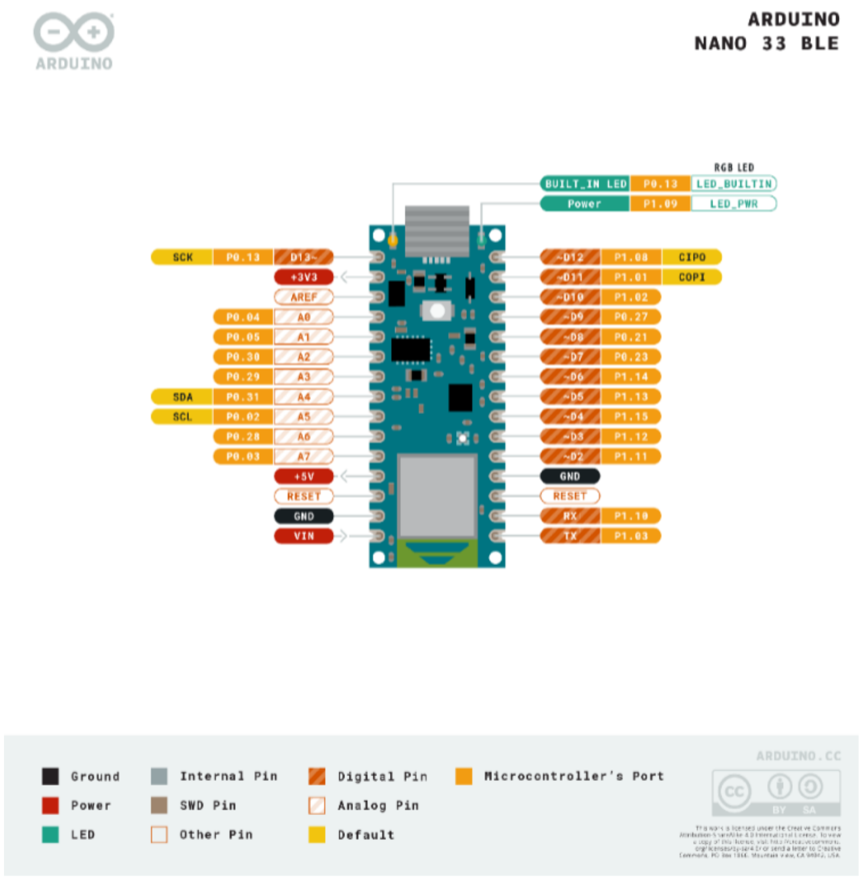

Figure 4: Arduino Nano 33 BLE pinout.

7.2 Timers

All timed actions must make use of the timer peripheral to set the trigger pulse, measure the duration of the received pulse, and trigger periodic acquisition of the values provided by the ADCs. To use the already developed abstraction layers related to these peripherals, it is possible to include the "timer.h" library or “TimerInterrupt_Generic.h” (included in the Arduino IDE).

However, if such functions are used, the internal mechanism of each must be explained in the report (e.g., what is the prescaler value, how counts are converted to time, etc.)

7.3 I2C

Arduino has a library called “Wire.h” that assists in the handling of serial communications. The I2C communication with the accelerometer must use this library, by implementing custom read and write functions. The internal mechanism of each function as well as the I2C registers interaction must also be explained in the report.

7.4 Processor Interrupts

Arduino has an abstraction layer that assists in handling interrupts. External interruptions handler can be set to trigger a function using the function *attachInterrupt()*.

Please note that the interrupt handlers run at priority level APP_LOW. The executed code must be reduced without applying complex operations. If other peripherals are accessed must use nRF52840 HAL functions or inline assembly.

7.5 Libraries

In addition to the libraries referred in the previous subsections, it is still allowed to use serial communication functions to send/receive information to/from the computer. Communication must be started in the setup function after setting the pins using *"Serial.begin(<baudrate>)"*. Common values for the baudrate are 9600 or 115200.

Any code section that uses open-source code or libraries (internal or imported) will be disregarded and will not be considered towards the final grade. All the interfaces and data acquisition, generation of interrupts, timers and conversions should be made by the authors. Advance functions such as *noTone()*, *pulseIn()*, *pulseInLong()*, *shiftIn()*, *shiftOut()*, *tone()* and Class Stream should not be used.

7.6 Calibration and complex operations

If calibrations are required, use the multi-segment curves approach: each segment is a linear regression and store the parameters in a lookup table.

The use of float and doubles variables should be minimized, favoring the use of variables in integer formats (e.g. uint8_t, int32_t, long) and, whenever possible, calculations in these formats without losing precision.

7.7 Some practical considerations

Start by testing the infrared sensor with a white paper sheet as it is a good reflective surface. The IR receiver cannot be used in direct sun light.

It is necessary to program some configurations/registers in the accelerometer before accessing the values.

It is suggested to use Arduino IDE offline version, installed on the computer.

Do not use the D0 and D1 pins (P1.10 and P1.03) since these are reserved for the serial communication with the computer.

The code loading is done automatically by a bootloader programmed in the Arduino module. However, the USB communication uses specific peripheral from the microcontroller's core. If the developed application affects the settings of this peripheral, the connection with the computer may be lost. In this case, Arduino will no longer appear as a serial port in the IDE. The loading of a new code is only possible by forcing the Arduino to stay in bootloader mode, avoiding running the bad code. This can be done by pressing the reset button twice (< 0.5 sec). The built-in led will initiate a fade-in fade-out sequence to signaling that it is in the bootloader mode.

8 Bonus Objective – IR Docking and Orientation Validation

As a bonus, students may implement **additional validation or algorithmic enhancements** that improve the reliability and usability of the pedometer system. Three possible extensions are proposed below.

- **IR Docking:** Use a short-range IR reflective pair (Vishay IR LED + phototransistor) to detect the presence of a nearby horizontal surface (e.g., the laptop lid). This confirms that the device is properly placed before operation and that the ultrasonic sensor faces forward in a known orientation.
- **Orientation Check:** Use IMU data to verify that the device's pitch and roll angles are within asmall tolerance (e.g., ±10°). This ensures that step counting based on the Z-axis is only enabled when the device is correctly aligned with gravity.
- **Orientation-Independent Step Detection** Extend the step-detection algorithm to operate correctly regardless of device orientation by combining acceleration data from all three axes.
- This can be achieved, for example, by computing the **magnitude of the acceleration vector** or by estimating and removing the gravity component. With this improvement, step counting becomes independent of how the device is held or mounted.

For the **IR Docking** and **Orientation Check** options, if placement or alignment conditions are not satisfied, the device should issue an appropriate warning (e.g., through Serial or BLE), such as:
"Device not docked — please place it in a horizontal position with the ultrasonic sensor facing forward to initiate measurement."

Note: Implement the infrared sensor system, applying biasing and amplification electronics [8] (discrete components). Convert sensor response to distance close/far classification to identify "docking".

Suggested keywords: IR reflective sensing, analog thresholding, hysteresis, tilt estimation, pitch/roll validation, startup check.

9 Bibliography

[1] Arduino, “Arduino General Website,” Arduino, [Online]. Available: https://www.arduino.cc/.

[2] Nordic, “nRF52840 Product Specifications,” Nordic, [Online]. Available: https://content.arduino.cc/assets/Nano_BLE_MCU-nRF52840_PS_v1.1.pdf.

[3] Arduino, “Arduino Nano 33 BLE product page,” [Online]. Available: https://docs.arduino.cc/hardware/nano-33-ble.

[4] ST, “LSM9DS1 9-axis iNEMO inertial module (IMU),” [Online]. Available: https://www.st.com/en/mems-and-sensors/lsm9ds1.html.

[5] A. Electronics, “Description of the HC-SR04 Ultrasonic Sensor,” [Online]. Available: https://ampere-electronics.com/p/hc-sr04-ultrasonic-sensor-module-3/.

[6] Vishay, “High Speed Infrared Emitting Diode, 890 nm,” [Online]. Available: https://www.vishay.com/docs/81040/tssf4500.pdf.

[7] Vishay, “Silicon PIN Photodiode TEFD4300,” [Online]. Available: https://www.vishay.com/docs/83471/tefd4300.pdf.

[8] L. Orozco, “Optimizing Precision Photodiode Sensor Circuit Design,” Analog Devices, [Online]. Available: https://www.analog.com/en/technical-articles/optimizing-precision-photodiode-sensor-circuit-design.html.