SMARTWATCH PROTOTYPE

Enzo Gonçalves Pusiol
Humberto Barbosa Coutinho
Vinícius Franklin Pedroso Mansur de Azevedo
Juiz de Fora
2025
Faculty of Electrical Engineering
Embedded Software

INTRODUCTION

In recent years, the appeal of using remote devices such as smartwatches has become increasingly popular. This factor is driven both by the advancement in embedded technologies - which operate through more precise and low-power sensors, along with the possibility of connectivity with other devices, making it an essential tool for any audience - and by the need to monitor health and physical performance.

In this context, these devices are generally structured to provide information such as step counting, distance measurement, calories expended, altitude oscillations, and characteristics like temperature and humidity.

Therefore, given this scenario, this work presents the development of a prototype similar to a smartwatch, capable of measuring distance, steps, speed, calories burned, altitude, temperature, and humidity. The project explores the application of embedded software to ensure the efficient and accessible collection, processing, and presentation of this information to the user.

SENSORS USED

GPS Module

The main function of the GPS module used in the project is to provide geographic location data for calculating the distance traveled, speed, altitude, time, and calories burned by the user. Communication with the microcontroller occurs via UART, and the received data is processed to extract information such as latitude, longitude, and time.

This report details the implementation of the GPS code, explaining how the altitude delta, the latitude and longitude difference, the calculation of the distance traveled using the Haversine Formula, and the estimation of calories burned are performed. Furthermore, we will detail the calculations of average and instantaneous speed.

Obtaining the Altitude Delta

The altitude variation is an important parameter to determine the user's physical exertion. Ascents and descents directly influence calorie consumption. In the code, the altitude is extracted from the GPS readings and stored for comparison with the previous reading.

The altitude delta is obtained as follows:
1. The code maintains a variable that stores the last recorded altitude.
2. When a new GPS reading is received, the current altitude is compared with the previous one.
3. If there is a significant difference, this value is stored and used for later analysis.

This approach allows for a more accurate estimation of the terrain's impact on the user's performance.

Calculating the Latitude and Longitude Difference

The latitude and longitude values are extracted from the NMEA sentences sent by the GPS. These sentences contain location information in decimal format and need to be processed for distance calculations.

In the code, the latitude and longitude values are stored in variables for subsequent calculations. The toRadians() function converts the degree values to radians, preparing them for the necessary trigonometric calculations.

The difference between the coordinates of the two points will be fundamental to determining the distance traveled, which will be detailed in the next section.

Calculating Distance Using the Haversine Formula
The Haversine Formula is used to calculate the distance between two points on the Earth's surface, considering the planet's curvature.

The distance calculation uses the Haversine Formula, which involves converting latitude and longitude coordinates to radians, calculating the angular differences, and applying trigonometric functions to determine the distance between the points. The result is multiplied by the Earth's average radius to obtain the distance in kilometers.

The Haversine Formula is described by:
$d = 2R * \arctan(\sqrt{a}, \sqrt{1-a})$
where:
$a = \sin^2((\Delta\varphi)/2) + \cos(\varphi1) \cos(\varphi2) \sin^2((\Delta\lambda)/2)$
and R is the Earth's average radius (6371 km).

Explanation of the Formula
1. Conversion to radians: The latitude and longitude coordinates are converted to radians.
2. Calculation of angular difference: The variations dLat and dLon are determined.
3. Calculation of the haversine factor: The equation uses the sin() function to calculate a factor that measures the separation between the points.
4. Application of the formula: The result is multiplied by the Earth's average radius (6371 km) to obtain the actual distance in kilometers.
5. This method is widely used by navigation systems due to its accuracy and low computational cost.

Average Speed
The average speed is calculated at the end of the journey by dividing the total distance traveled by the total time spent. The calculation is performed using the start and end times of the journey, which are stored at the beginning and end of the run, respectively.

Instantaneous Speed
The instantaneous speed is calculated with each new GPS reading by dividing the distance between the two most recent points by the time elapsed between these readings. This calculation is performed using the latitude and longitude coordinates and the reading times of the most recent points.

Total Distance Traveled
The total distance traveled is accumulated with each new GPS reading by adding the distance calculated between the most recent points. This sum is performed using the Haversine Formula to calculate the distance between the points and accumulating the result in a variable that stores the total distance traveled.

Calculation of Calories Burned
The estimation of calories burned takes into account the distance traveled, the average speed, and the user's weight. Based on the website:

https://sites.google.com/site/compendiumofphysicalactivities/

Activity-Categories/running

And the MET equation was based on:

https://nutritotal.com.br/pro/material/met-multiplos-de-equivalentes-metabolicos/

The calculation in the code is based on determining the MET (Metabolic Equivalent of Task) based on the user's speed and applying a formula that takes into account the MET, the user's weight, and the exercise time.

Explanation of the Calculation
1. Determination of MET (Metabolic Equivalent of Task): MET is a factor that represents calorie expenditure based on the type of exercise.
2. Calorie formula: MET is multiplied by the user's weight and the exercise time in hours.
3. Energy adjustment: The factor 1.05 represents an average of the extra expenditure due to metabolism variations.

This approach allows for a reasonably accurate estimation of calories burned during physical activity.

Uncertainties of the GPS Module and Error Limitation

It is important to note that the GPS module presents some uncertainties, especially when the user is at rest. Small variations in GPS readings can result in fluctuations in both distance and altitude, generating false values.

To mitigate these problems, thresholds were established in the code. These thresholds help filter out insignificant variations, ensuring that only significant changes are considered in the distance and altitude calculations. Specifically:

* Altitude Variation: Small variations in altitude, less than 0.3 meters, are ignored. This prevents irrelevant fluctuations from affecting the calculation of the altitude delta.
* Distance Variation: Distances less than 0.003 kilometers (3 meters) between two consecutive readings are also disregarded. This prevents small variations in GPS position when the user is stationary from being interpreted as real movement.

These thresholds were defined after tests and calibrations to ensure that the system provides accurate and reliable data, even in conditions where the GPS might exhibit small variations.

With these calibrations, the system becomes more robust and accurate, providing more reliable data for the calculations of distance, altitude, and calories burned.

Conclusion
The implementation of the GPS module in the embedded system allows for real-time monitoring of the user's location and the calculation of important metrics such as distance traveled, speed, and calories burned.

Obtaining the altitude delta provides information about terrain variation, directly influencing the calculation of physical exertion. The calculation of the latitude and longitude difference, combined with the Haversine Formula, ensures accurate measurements of the distance traveled.

The estimation of calories allows the user to track their performance and adjust their activity routine. In the future, improvements can be made in the calibration of the sensors and the optimization of energy consumption, increasing the device's autonomy.

ADXL345 Accelerometer

The ADXL345 is a low-power digital accelerometer designed to measure acceleration in three axes (X, Y, and Z) with high precision. It is used in remote devices, such as smartwatches and activity trackers, due to its ability to detect movements, tilt, and falls.

Technical Characteristics
The ADXL345 operates in a voltage range of 2.0 V to 3.6 V and can be connected via I2C or SPI, allowing flexible integration with microcontrollers and embedded systems. Its main features include:

* Adjustable measurement range: ±2g, ±4g, ±8g, or ±16g;
* High resolution: Sensitivity up to 4 mg/LSB in high-resolution mode (±2g);
* Configurable sampling rate;
* Free-fall and motion detection sensors;
* Low power consumption: Only 40 µA in normal operating mode and 0.1 µA in standby mode.

Application in Smartwatches
In the context of the proposed device, the ADXL345 is used for step counting: the accelerometer detects characteristic oscillations of walking and running to calculate the number of steps taken by the user.
The integration of the ADXL345 with the device's embedded software requires the implementation of algorithms to manipulate the readings, identify variations, and eliminate unwanted noise. The use of a moving average helps to smooth out abrupt variations in the collected data.

Code Implementation
The sensor is initialized in a function where the power control register is configured to activate the device.
The reading of the X, Y, and Z axis values occurs through the adxl345_read() function, which accesses the register and retrieves 6 bytes of data, corresponding to the three axes.
The task_adxl345() function is responsible for processing the raw sensor data. It calculates the magnitude of the acceleration using the equation:

Magnitude = sqrt(x^2 + y^2 + z^2)

In addition, a moving average filter is applied to smooth the acceleration values and reduce noise. Finally, the magnitude is compared with a step detection threshold (STEP_THRESHOLD = 280) and the step counter is incremented.

OLED Display

The OLED (Organic Light-Emitting Diode) display is a technology widely used for device screens, being extensively employed in cell phones and smartwatches due to its energy efficiency, high contrast, and excellent visibility under different lighting conditions. Unlike traditional LCD screens, OLED displays do not require a backlight panel, as each pixel emits its own light, resulting in sharper images.

Technical Characteristics
OLED displays used in embedded systems typically have the following characteristics:

* Low power consumption: As black pixels consume no energy, consumption is reduced, making OLED ideal for portable devices;
* Versatile communication interfaces: Can operate via I2C or SPI, allowing easy integration with microcontrollers and embedded processors;
* Variable resolution and sizes.

Application in the Smartwatch

In the device under development, the OLED Display plays a fundamental role in displaying the information collected by the embedded sensors. Its main applications include:

* Real-time data presentation: Display of steps, distance traveled, speed, calories burned, temperature, humidity, and altitude;
* Interactive interface: Interactive screens for the user.

The implementation of the OLED Display in the smartwatch's embedded software requires the use of specific libraries for graphic control.

Code Implementation
The display initialization occurs in the ssd1306_init() function, which configures the charge pump mode, screen orientation, and display activation. The ssd1306_clear_display() function is used to clear the displayed content, and text display occurs in ssd1306_display_text(), which:
* Iterates through the string to be displayed and writes the characters one by one using the font8x8_basic font as a dictionary.
* Supports multiple lines and advances to the next line upon encountering a newline character (\n).

The task_ssd1306_display_text() function controls the user interface, allowing switching between different screens by pressing a button (BT_IO):
* Screen 0: Displays the step count collected from the ADXL345.
* Screen 1: Displays the temperature and humidity collected from the DHT11 sensor.

This interface ensures that the user can view relevant information dynamically.

DHT11

The DHT11 sensor is a widely used device for measuring air temperature and relative humidity in embedded systems. It communicates with the microcontroller through a single digital pin, sending discrete readings every second. In the context of this project, it is used to collect and display environmental information on the SSD1306 OLED Display. The following will detail the implementation of the DHT11, including initialization, data reading, and the display of the collected information.

Code Implementation
Communication with the DHT11 occurs through the vTaskGet() function, which executes a continuous loop to capture and process sensor data. First, the function initiates an activity log to indicate that measurement is in progress. Then, the temperature and humidity values are read using the DHT11_read() function, which returns a structure containing the collected readings. These values are stored in the global variables temp and umi, allowing other parts of the program to access and use the information. After each reading, the data is sent to the ESP32 log, ensuring continuous real-time monitoring of temperature and humidity.

The code provides for the display of temperature readings on Screen 1 of the OLED display, allowing switching between different data by pressing a button. The periodic collection of DHT11 data enables environmental monitoring, making it a useful tool for physical activities that require thermal control or humidity analysis.

CONCLUSION

The development of the smartwatch prototype demonstrated the feasibility of applying embedded systems for monitoring physical activities. The integration of sensors, such as the GPS, ADXL345 accelerometer, OLED display, and DHT11 sensor, allowed for obtaining data on distance traveled, speed, calories burned, step count, altitude, temperature, and humidity.

The implementation of algorithms, including the Haversine Formula for distance calculation and the estimation of calorie expenditure, ensured essential measurements for the device's usability. Furthermore, the use of FreeRTOS enabled the execution of tasks responsible for capturing and displaying information.

Despite the positive results, some improvements can be explored in the future, such as optimizing energy consumption, more precise calibration of sensors, and the implementation of new functionalities, such as heart rate monitoring. In this way, the project can evolve into an even more useful device for tracking users' health and physical performance.

BIBLIOGRAPHY

[1] AOSONG ELECTRONICS. DHT11 Humidity & Temperature Sensor Datasheet. Available at: https://akizukidenshi.com/download/ds/aosong/DHT11.pdf. Accessed on: Mar. 16, 2025.
[2] ANALOG DEVICES. ADXL345 Datasheet: 3-Axis, ±2 g/±4 g/±8 g/±16 g Digital Accelerometer. Available at: https://www.analog.com/media/en/technical-documentation/data-sheets/adxl345.pdf. Accessed on: Mar. 17, 2025.
[3] FREE RTOS. FreeRTOS – Real-time operating system for microcontrollers & microprocessors. Available at: https://www.freertos.org/. Accessed on: Mar. 17, 2025.
[4] YANBE. SSD1306 ESP-IDF I2C Driver. Available at: https://github.com/yanbe/ssd1306-esp-idf-i2c. Accessed on: Mar. 17, 2025.