Universidad Nacional de Córdoba
Facultad de Ciencias Exactas, Físicas y Naturales

Tomas Sarquis

Diciembre 2020

Práctica Supervisada
Segundas cien horas

Monitoreo del movimiento de animales vacunos

Supervisor: Gustavo Wolfmann

Índice

1. Introducción

El presente documento detalla la realización de la práctica profesional supervisada llevada a cabo por el estudiante Tomas Sarquis, perteneciente a la carrera de Ingeniería en Computación.

La práctica profesional tuvo una duración de 200 (doscientas) horas, repartidas en 41 (cuarenta y uno) días. Este documento detalla la segunda mitad de la misma.

En la documentación de la primera mitad de la práctica se ha comentando acerca del objetivo de la misma y del entorno de trabajo, por lo tanto, estos tópicos serán obviados en el presente documento.

2. Descripción del trabajo

2.1. Investigación

Si bien la etapa anterior de investigación abarcaba todas las necesidades, se tuvo que replantear ciertas cosas, por lo que la en la segunda parte se hizo inminente.

Una de estas cosas fue el guardado en memoria de las mediciones. Después de muchas idas y vueltas (en cuanto a qué tipo de memoria usar) se buscó respuestas en la comunidad, quién respondió satisfactoriamente. [1]

2.2. Desarrollo

Habiendo terminado de configurar el sensor y de haber podido recibir de manera exitosa las primeras mediciones de aceleración, lo siguiente era el guardado de las mismas y el posterior envío.

2.2.1. Almacenamiento

El microcontrolador cuenta con distintas maneras de almacenar datos.

Primeramente se optó por utilizar la memoria *NVS*[2]("Almacenamiento no volátil") por su facilidad, pero se tuvo que pensar en otra manera debido a que este método utliza la memoria *Flash*, y el uso constante de la misma tiende a degradarla.

Finalmente, con ayuda del supervisor de práctica, se decidió por usar un sistema (figura 1) de memorias de dos niveles: los datos recibidos desde el sensor se van acumulando en la memoria *RTC RAM* del microcontrolador (que no se degrada con el uso), y cuando ésta se llena, se transfieren los datos al *filesystem SPIFFS* (que tiene muchas más capacidad de almacenamiento), dejando vacía la primera. Luego se vuelve a usar la *RTC RAM*, y el proceso se repite hasta que el *filesystem* agota su capacidad.

El sistema *SPIFFS* utliza la memoria *Flash*, pero como está en un "segundo nivel", no se escribe tan seguido y por lo tanto, no se degrada tan rápidamente.

Cuando el *filesystem* se llena, se debe vaciar y transferir los datos al servidor.

Se busca que el envío de las mediciones no se realice de manera frecuente, ya que el consumo de energía se vería afectado negativamente. Es por esto que se intentó aprovechar al máximo la fase de almacenamiento.

2.2.2. Transferencia

La idea es que, cuando sea necesario, se pueda enviar todas las mediciones a un servidor de la red local (en un *drone* que circula cerca) de forma segura.

[1] `www.stackoverflow.com/questions/63780850/esp32-best-way-to-store-data-frequently`

[2] `docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/storage/nvs_flash.html`

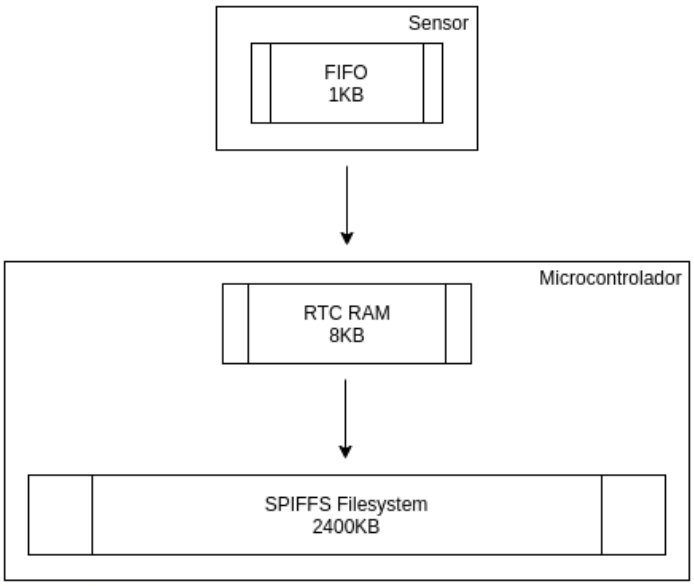

Figura 1: Esquema de memoria en niveles

Para esto, se usó el protocolo *HTTP*, mientras que la tecnología inalámbrica fue *Wi-Fi*[3].

La transferencia consiste en varios paquetes *HTTP* de tamaño configurable. Para las pruebas se han usado paquetes de 100 datos cada uno.

2.2.3. FreeRTOS

El sistema operativo que orquesta el sistema es *FreeRTOS*, el cuál estuvo diseñado por 3 tareas:

- **Tarea principal:** Encargada de crear las demás tareas y los recursos compartidos.
- **Tarea de acelerómetro:** Su misión es la de recepción y almacenamiento de las mediciones.
- **Tarea de transferencia:** Realiza las tareas de envío de datos al servidor.

2.3. Resultados

Los resultados a continuación detallados han sido recavados de la ejecución del sistema en el *Intelytrace* y no en la placa de pruebas.

[3]El sistema debía usar *LoRa* pero el estudiante no contó con el *hardware* necesario

Figura 2: *Drone* funcionando como servidor

2.3.1. Mediciones

Los resultados arrojados se pueden observar en la figura 3, los cuales fueron obtenidos midiendo la aceleración humana caminando y trotando a distintas velocidades. La cantidad de mediciones es de 1488.

2.3.2. Consumo

Uno de los requisitos principales era que el consumo de la placa sea lo menor posible y para esto se hizo incapié en que el microcontrolador se encuentre en modo *sleep* la mayor parte del tiempo.[4]

A pesar de esto, los resultados obtenidos en cuanto al amperaje no fueron los esperados, como se puede ver en la tabla 1.

Cabe aclarar que la corriente eléctrica (consumo) fue medida en serie a la batería que alimenta toda la placa.

Consumo esperado	Consumo obtenido
10 [uA]	4.3 [mA]

Cuadro 1: Consumo esperado y obtenido (*deep-sleep mode*)

[4]El estudio del consumo considera únicamente las etapas de *sleep* ya que en ésta etapa transcurre más del 90 % del programa

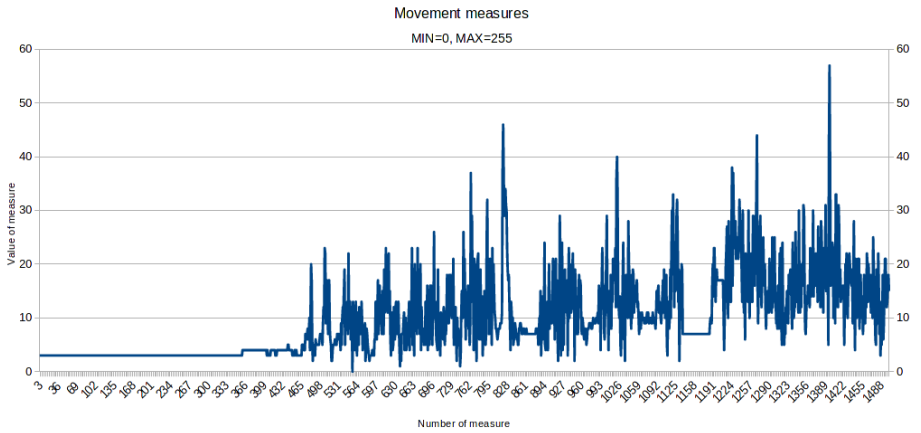

Figura 3: Resultados obtenidos

Si bien no se sabe, a ciencia cierta, a qué se debe esta diferencia de consumo, la misma no se atribuye a un problema del *software* implementado, ya que se han llevado a cabo pruebas con programas oficiales[5] y los resultados han sido los mismos.

Por su parte, la placa incrustada en el *Intelytrace* no solo cuenta con el ESP32, si no que cuenta, además, con otros componentes de *hardware* como cargador de batería, receptor LoRa WAN y GPS, por lo que se plantea como hipótesis que dichos componentes sean los causantes del alto consumo, y no el ESP32 propiamente dicho.

[5] `www.github.com/espressif/esp-idf/tree/master/examples/system/deep_sleep`

3. Conclusión

La práctica supervisada ha sido una actividad realmente enriquecedora tanto para el alumno como para el supervisor y su empresa. El estudiante tuvo una experiencia real de la implementación completa de un *software*, pasando por el análisis de los requerimientos, el diseño, las posibilidades de desarrollo, las investigaciones posteriores y finalmente el desarrollo y los resultados.

El sistema implementado en estos meses es susceptible a (y debería, en lo posible) ser mejorado, ya que ciertas cuestiones del programa lo ameritan. Además, se recomienda hacer más pruebas de *testing*, ya que el desarrollador pudo hacer solamente las pruebas básicas.

Algunas mejoras podrían ser:

- Reemplazar la conexión WiFi por LoRa WAN
- Tener en cuenta valores de movimiento relativos. Esto es, no solo tener en cuenta los valores absolutos, si no, además, computar las diferencias de movimiento en función del tiempo. Esto podría ser útil para detectar cambios bruscos de movimiento.
- Arreglo de *bugs*

A pesar de esto y teniendo en cuenta el tiempo empleado para el desarrollo y la poca experiencia del desarrollador, el resultado es muy favorable: los requerimientos fundamentales han sido satisfechos.

Los conocimientos obtenidos en la universidad y las ayudas recibidas por parte del supervisor y de la comunidad informática, fueron de necesidad para el estudiante.

4. Apéndice

El código del programa implementado se puede apreciar en el repositorio[6] del desarrollador. Los archivos de interés se encuentran sobre el directorio *esp-idf/main* que, a su vez, contiene tres directorios más:

- *include*. Contiene los *headers* del código fuente.
- *lib*. Dentro se encuentran las librerías externas que han sido usadas.
- *src*. El código fuente de cada tarea se encuentra aquí.

El área de mayor interés es, quizás, la del directorio *src*. En él se encuentran implementadas las tareas del sistema, por lo que para conocer su funcionamiento, es de crucial importancia entender estos códigos.

[6] `www.github.com/tsarquis88/esp32_freertos`