Microcontroladores y
Electrónica de Potencia

Trabajo integrador:

GPS TRACKER

Alumnos:

Bruno, Simón - 12177

Cárdenas, Facundo - 11652

Introducción

La idea propuesta surge de una situación muy frecuente y molesta que todos conocemos, y es cuando no para de sonar la alarma de algún auto en la calle y el dueño nunca se entera. Es por esto que se trata de simular una aplicación en la cual cuando suene la alarma del auto, se envíe información mediante mensaje de texto a algún dispositivo celular para notificar al dueño del vehículo y proveer la ubicación en tiempo real del auto.

Se propone hacer una aplicación la cual pueda:

- Obtener la información de posición a través de la comunicación UART del módulo de gps con el microcontrolador.
- Enviar un mensaje de texto con la ubicación a un teléfono celular mediante un módulo gprs.
- Configurar y depurar la aplicación mediante una PC.

Esquema tecnológico

En el siguiente diagrama de bloques se representan los módulos del sistema y los sistemas externos que interactúan con él.

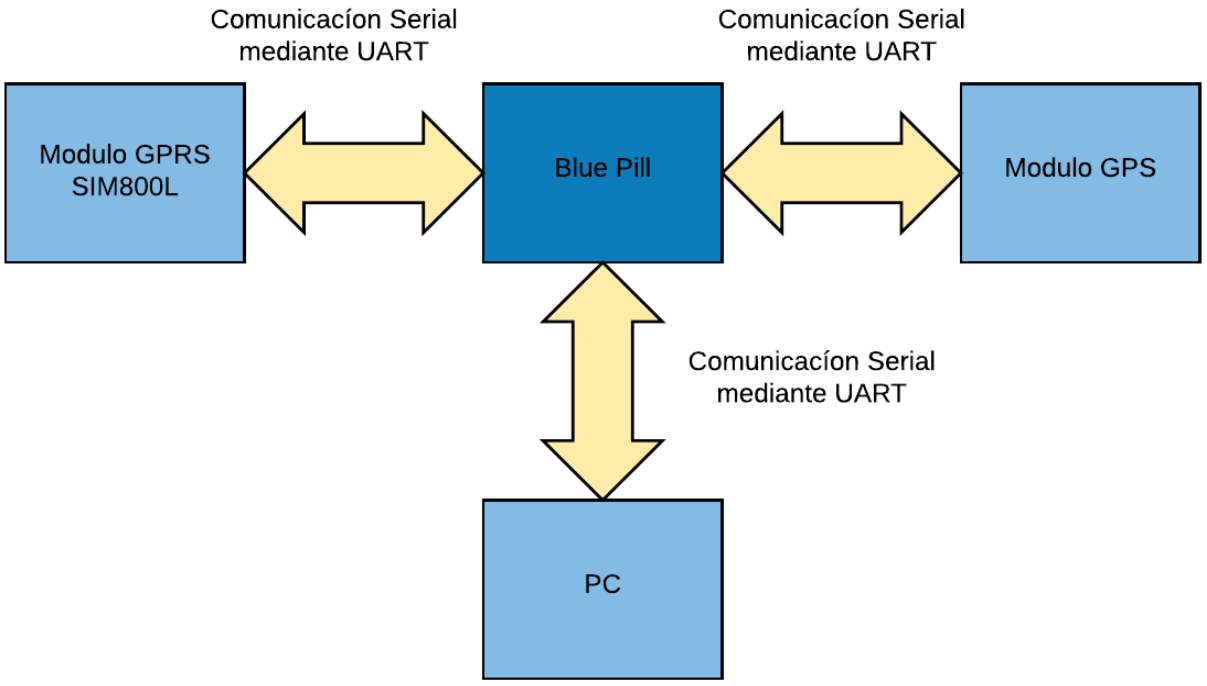

Figura (1) Diagrama de bloques del sistema

Se realizó el esquemático de las conexiones utilizando Fritzing, adaptando los módulos ya cargados. Las conexiones son muy simples, pero sirve para ver cómo están conectados los módulos a la BluePill. Todos los pines que transmiten señal al microcontrolador son de color verde, y los que reciben, son grises. Vcc y GND son rojos y negros, respectivamente.

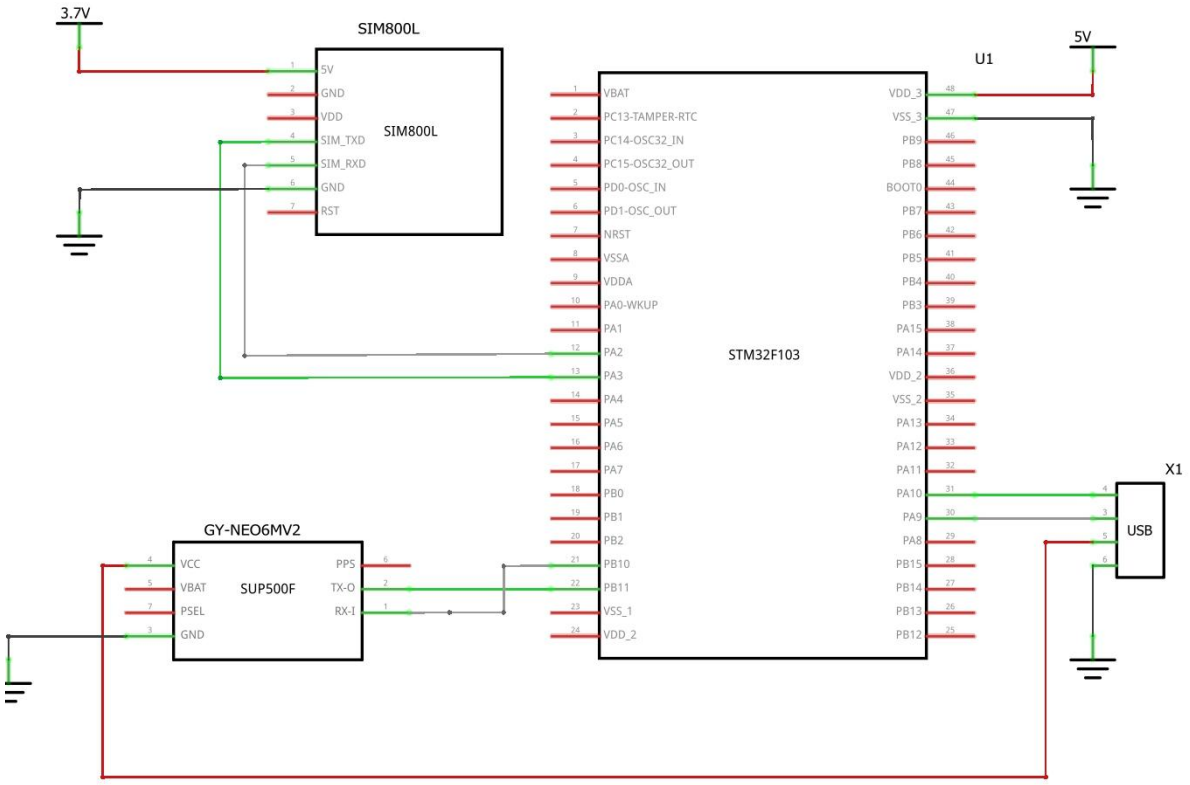

Figura (2) Esquemático de conexiones

Detalle de módulos

Para desarrollar el proyecto se utiliza una placa Blue Pill basada en un microcontrolador STM32F103, que cuenta con 3 interfaces de comunicación serie asíncrona UART, mediante las cuales se comunica la placa con los módulos y sistemas externos a utilizar.

Módulos

- Módulo GSM/GPRS SIM800L

GSM es un sistema de comunicación global para los teléfonos móviles y es un tipo de red que se utiliza para la transmisión móvil de voz y datos, pero su ancho de banda es lento y a veces puede causar interferencias electrónicas. GPRS es una extensión mejorada del GSM, permitiendo velocidades de transferencia mayor y una cobertura inalámbrica completa. Este módulo utiliza un MicroSIM con crédito de alguna compañía telefónica para enviar y recibir llamadas y SMS y conectarse a Internet. Para configurarlo y controlarlo se hace mediante comandos AT.

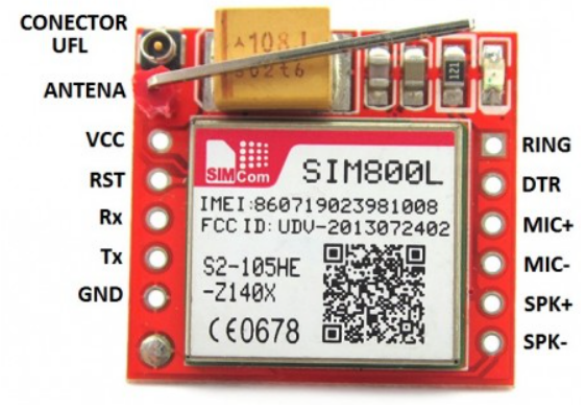

Imagen (1) SIM800L

- Módulo GPS Ublox Gy-neo6mv2

GPS consiste en una constelación de 24 satélites en órbita que transmiten información precisa de tiempo y posición (órbita del satélite) a estaciones terrestres, a través de mensajes de Latitud, Longitud, Velocidad, Rumbo y Tiempo. Para que el módulo funcione correctamente debe conectarse como mínimo a 4 satélites. Su funcionamiento normal utiliza el protocolo NMEA (National Marine Electronics Association) pero en el proyecto no se utilizó y se reemplazó por un polling de la información en el momento específico.

Imagen (2) GPS

El mensaje utilizado es la respuesta a una pregunta hecha al módulo a través del comando de propietario *$PUBX,00*33*, que nos brinda información de la posición a un horario específico, entre otras cosas. Esta sentencia tiene estructura similar a la utilizada por NMEA, pero es específica de la marca del GPS utilizado.

Message Structure:

```
$PUBX,00,hhmmss.ss,Latitude,N,Longitude,E,AltRef,NavStat,Hacc,Vacc,SOG,COG,Vvel,ageC,HDOP,VDOP,TDOP
,GU,RU,DR,*cs<CR><LF>
```

Figura (3) Estructura de mensaje

- Microcontrolador STM32F103C8T6 - BluePill

El STM32F103C8T6 montado en la Blue Pill es un microcontrolador de 32 bits con arquitectura ARM y un núcleo tipo cortex M3 con una frecuencia máxima de trabajo de 72 MHz. Incluye 20 KB de memoria RAM y 64 KB de memoria Flash. La mayor utilidad en el proyecto es la posibilidad de utilizar 3 puertos de conexión UART y su reducido tamaño.

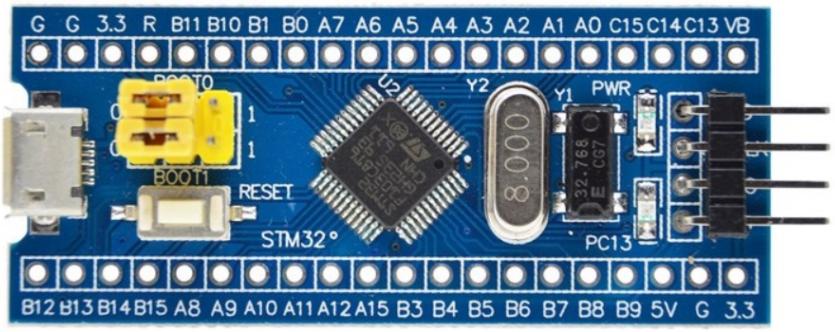

Imagen (3) BluePill

Sistemas Externos

- PC: Mediante un terminal serial se envían tramas a la placa para configuración y depuración. Esto no es parte de un funcionamiento real, pero aporta a la comprensión de los mensajes enviados y recibidos.

Funcionamiento general

Basándose en el esquema tecnológico anterior, graficado en las Figuras (1) y (2), se puede describir el funcionamiento general de la aplicación. Inicialmente se enciende la aplicación y espera a que se active la alarma, la cual se simula ingresando una trama correspondiente con un terminal serie en una PC. Una vez ingresado dicho mensaje, se procede a obtener la ubicación actual del dispositivo mediante el módulo GPS. El mensaje recibido tiene varios datos que tendremos que decodificar para poder leer la información limpia y poder interpretarla correctamente. Una vez obtenida la ubicación y hora, se envía un mensaje de texto con el módulo GPRS SIM800L al número de teléfono correspondiente. En el siguiente esquema se observa lo explicado anteriormente.

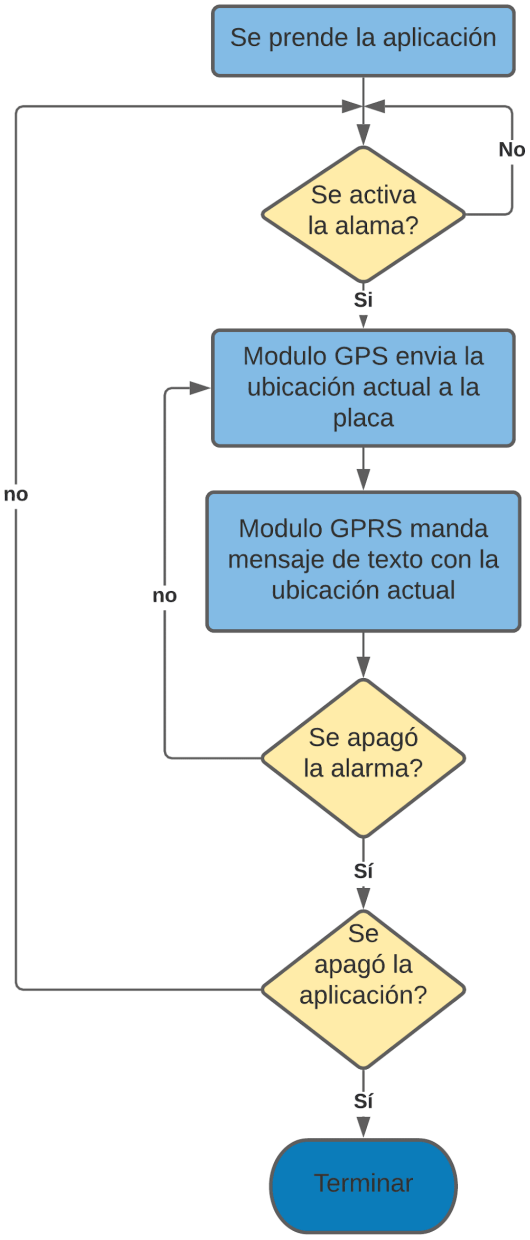

Figura (4) Funcionamiento general

En la imagen siguiente se observa el funcionamiento general de la aplicación desde el punto de vista del usuario, en donde nos llega un mensaje de texto con la activación de la alarma y luego cada 10 segundos llegan los mensajes con la hora en formato UTC (Coordinated Universal Time), la latitud y longitud, de la forma: Latitud/Longitud: grados minutos.fracción de minuto.

Imagen (4) Mensajes recibidos por la Aplicación

Programación

El proyecto comenzó con la configuración del microcontrolador utilizando la interfaz MX del programa STM32CubeIDE para activar los terminales de comunicación necesarios y el modo de operación del timer, con las respectivas interrupciones. Dejamos las 3 UART con la configuración por defecto, habilitamos interrupcion y el BAUD rate de 9600. Se configura un timer para mandar los mensajes cada 10 segundos (que luego configuraremos por consola). Los tiempos elegidos son para hacer más práctica la demostración de funcionamiento de la aplicación. Se utiliza el Timer 2 de 16 bits usando el cristal interno del micro configurando el reloj para trabajar a 1 Mhz. Configurando el Prescaler en 1000 y el Counter Period o Tope a 9999 (10000-1) se llegan a los 10 segundos deseados: $\frac{1Mhz}{Prescaler} * Counter\ Period = 10s$. Luego cambiando el Prescaler podremos cambiar el tiempo.

La conexión elegida es la que se puede ver en la Figura (2), donde se puede ver que la UART1 se utilizó para la comunicación con la PC, la UART2 para la comunicación con el SIM800L y la UART3 para el GPS.

El funcionamiento básico del código es a través de las rutinas de interrupción de los puertos UART y del timer, que llamarán a las funciones de procesamiento de datos (*text_msn*, *interpreta_gps* e *interpreta*) para obtener los strings que contienen los datos a enviar por SMS. El código *main* se encarga únicamente de enviar los mensajes de apagado de las tramas de GPS no utilizadas y de activación de interrupciones. En las imágenes siguientes

haremos una breve descripción de las funciones más importantes del código para comprender más fácilmente su aplicación.

```
423 /* USER CODE BEGIN 4 */
424 void text_msn(mensaje){
444
445 void interpreta_gps(void){
525 void interpreta(void)
584 void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
615
616 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim){
620
621 /* USER CODE END 4 */
```

Figura (5) Funciones implicadas en el procesamiento y envío de de los mensajes correspondientes.

En la Figura (5) se puede ver las funciones donde está condensado el funcionamiento general del código. Cuando se envíe algún comando por consola, será decodificado en la rutina de interrupción de la UART1, ubicado en la función *HAL_UART_RxCpltCallback()* y luego redirigido a la subrutina dentro de *interpreta.* Luego, según lo requiera cada uno de los posibles funcionamientos, se consigue la posición del vehículo en *interpreta_gps* y es enviado al celular en la función *text_msn.*

```
525 void interpreta(void)
526       {
527     uint16_t nuevot;
528     nuevot=0;
529         switch(dato_Rx[0])
530             {
531             case 'g':
532             case 'G':  // :G trama delGPS
533                 if(strstr(dato_Rx, "ubicacion") || strstr(dato_Rx, "UBICACION") ){
534                     interpreta_gps();
535                     // hacemos pedido de datos al gps
536                 }
537                 memset(dato_Rx, 0, SIZE_RX);
538                 break;
```

Figura (6) Ejemplo de interpretación de comando

La interpretación de los distintos comandos ingresados por consola se realiza en la función *interpreta*, donde se determinan las tareas a realizar según sea el “case” correspondiente . Los casos contemplados son los de solicitud de la posición, configuración del timer según 3 valores posibles, encendido/apagado de la alarma y un *help* de los distintos comandos.

```
452     for (char *pV = strtok(buff, ","); pV != NULL; pV = strtok(NULL, ",")) {
453         switch (cnt) {
454         case 2:
455             utcRaw = strdup(pV);
456             break;
457         case 3:
458             latRaw = strdup(pV);
459             break;
460         case 4:
461             hemNS = strdup(pV);
462             break;
463         case 5:
464             lonRaw = strdup(pV);
465             break;
466         case 6:
467             hemEW = strdup(pV);
468             break;
469         }
470         cnt++;
471     }
```

Figura (7) Extracción de argumentos de GPS

Dentro de la función *interpreta_gps,* el núcleo del funcionamiento se encuentra en la subrutina *for* de la Figura (6), que, a través de la función *strtok,* se encargará de guardar los distintos argumentos del mensaje obtenido como respuesta del comando *$PUBX,00*33* en las respectivas variables, que luego conformarán el mensaje final a enviar al celular.

```
616 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim){
617         interpreta_gps();
618         // pedido y procesamiento de los datos del gps cada vez que se cumple el tiempo
619 }
```

Figura (8) Rutina interrupción del Timer

Una vez alcanzado el tope de conteo del timer, el código realiza la rutina de interrupción respectiva, realizando un llamado a la función *interpreta_gps* para conseguir un valor de posición y enviarlo al teléfono.

```
424 void text_msn(mensaje){
425     HAL_UART_Transmit(&huart1,(uint8_t *)AT,strlen(AT), HAL_MAX_DELAY);//Probamos el modulo
426     HAL_UART_Transmit(&huart2,(uint8_t *)AT,strlen(AT), HAL_MAX_DELAY);//Probamos el modulo
427
428
429     HAL_UART_Transmit(&huart1, (uint8_t *)ModoM,strlen(ModoM), HAL_MAX_DELAY);//Ponemos el modulo en modo mensaje
430     HAL_UART_Transmit(&huart2, (uint8_t *)ModoM,strlen(ModoM), HAL_MAX_DELAY);//Ponemos el modulo en modo mensaje
431
432
433     HAL_UART_Transmit(&huart1, (uint8_t *)Number,strlen(Number), HAL_MAX_DELAY);//numero al que le mandamos mensaje
434     HAL_UART_Transmit(&huart2, (uint8_t *)Number,strlen(Number), HAL_MAX_DELAY);//numero al que le mandamos mensaje
435
436
437     HAL_UART_Transmit(&huart1, (uint8_t *)mensaje,strlen(mensaje), HAL_MAX_DELAY);//mensaje que mandamos
438     HAL_UART_Transmit(&huart2, (uint8_t *)mensaje,strlen(mensaje), HAL_MAX_DELAY);//mensaje que mandamos
439
440
441     HAL_UART_Transmit(&huart2, &ctrlZin, 1, 100);
442 }
```

Figura (9) Envío de comandos AT

Por último la función *text_msn*(mensaje) recibe como parámetro el mensaje que tiene que enviar por mensaje de texto(alarma encendida, ubicación y hora) y lo manda a la PC (para verificar que es correcto) y al módulo SIM800L. Para enviar un mensaje de texto mediante el módulo SIM800L se utilizan los comandos AT. Primero se tiene que poner el módulo en modo mensaje con el comando *AT+CMGF=1,* luego se debe ingresar el número de teléfono al cual mandar el mensaje con el comando *AT+CMGS=\"+54261xxxxxxx\"* (reemplazar x con el número) y ingresa el mensaje a enviar que fue recibido como parámetro, finalmente

para enviar el mensaje el módulo necesita que se envíe el fin de línea con “Ctrl + z” el cual no se puede enviar directamente por UART por lo que se envió el comando en hexadecimal 0x1A.

Etapas de montaje y ensayos realizados

El proyecto comenzó por la verificación del funcionamiento electrónico de todos los componentes, junto con los códigos de prueba y la resolución de los problemas que surgieron. Esta etapa tuvo algunas complicaciones por un módulo quemado y un cable de comunicación serie que tenía fallas de fábrica.

Una vez resuelta la electrónica de cada módulo, procedimos a la comunicación particular con cada uno de los circuitos, para aprender sobre los comandos, mensajes, limitaciones y posibles configuraciones. Esto se realizó usando el puerto serie de la computadora y un cable serie-TTL, enviando comandos específicos por teclado. Gracias a esto, pudimos comprender más sobre las áreas que tendría que cubrir el código, y algunos problemas que podríamos llegar a encontrar. Además, fue muy útil para decidir y comprobar que mensajes será necesario incluir dentro del código.

Habiendo explorado varias configuraciones del módulo GPS y luego de algunas consultas con los profesores, llegamos a la conclusión de que sería mejor el método Polling descrito más arriba para el desarrollo del proyecto, de manera que tuvimos que ajustar algunos comandos y rutinas de interpretación de la respuesta del módulo.

El módulo GPRS presentó algunas complicaciones por necesitar un chip con crédito y una señal satelital constante y sin interferencia, dificultando la etapa de configuración y búsqueda de errores.

Por la naturaleza de nuestro proyecto de manipulación de mensajes según distintos protocolos y medios de transmisión, la etapa final del trabajo consistió en comunicar y coordinar efectivamente esos mensajes, tomando la señal del GPS procesada y enviarla por SMS a través del GPRS. Se incluyó un control por puerto serie con la computadora, para simular el funcionamiento de la alarma y el disparamiento del timer para la obtención de la ubicación.

Resultados, especificaciones finales

Los objetivos iniciales fueron alcanzados con éxito tras varias complicaciones, pero pudimos cumplir con la idea inicial y con las recomendaciones de la cátedra para minimizar el consumo de energía. El funcionamiento real de este proyecto implica una señal proveniente de la alarma del auto, cuestión que no pretendemos resolver para no inferir en problemas no relacionados al desarrollo de esta materia.

Todos los circuitos consumen entre 3.3 y 5V, pero hay que asegurarle un pico de 2A al GPRS, por lo que habría que usar reguladores en serie para poder utilizar la batería Li-Po con los 3 módulos, pero es recargable con un cable al usb del auto o al conector del encendedor.

El consumo de saldo del chip de celular también es un factor a considerar. El funcionamiento del código y el gasto de credito en enviar mensajes depende en gran medida del tiempo que este andando el auto, por lo que se puede recurrir a un plan de

mensajes de texto de alguna compañia de telefonia movil, calculando el gasto por minuto de saldo y el tiempo posible de movimiento del vehículo.

Una característica problemática de este proyecto es la fuerte dependencia de la calidad de la señal recibida, por lo que si el vehículo pasa por debajo de un puente o por dentro de un túnel, el GPS no enviará señal correctamente. Esto influye mucho en el correcto funcionamiento del código, ya que cuando algun modulo pierde señal, hay que esperar que se recupere para volver a lanzar el código. La calidad de los componentes usados también repercute en la precisión de la posición obtenida. Según el datasheet, el GPS provee señal de posición con precisión de 2.5 metros, pero por la interferencia con los edificios y el techo por encima del módulo, la señal varia. De la misma manera, el GPRS depende de la señal satelital que consiga para enviar los mensajes. Acá también se puede caer en el problema de la demora de enviar los mensajes o de que lleguen al celular, pero lo solucionamos adjuntando la hora exacta del mensaje de posición.

El programa está diseñado para que funcione autónomamente, notificando del lanzamiento de la alarma y la actualización de la posición. La comunicación con la PC es para verificar los comandos enviados y poder configurar el timer mientras funciona el código. Las distintas posibilidades de temporización están restringidas para simplificar el funcionamiento general del código, ya que en una aplicación real tampoco podríamos comunicarnos con el módulo a través de una PC.

Conclusiones. Ensayo de ingeniería de producto

Este trabajo tiene un objetivo muy concreto de solucionar la situación de la alarma sonando sin parar y sin que el dueño se entere. Nuestra propuesta incluye el envio temporizado de mensajes de texto, pero tampoco es una solución muy eficiente. Estar recibiendo coordenadas del auto cada 30 o 60 segundos tampoco es práctico, pero si es una buena forma de empezar a idear una solución. A lo largo del desarrollo del proyecto, nos empezamos a dar cuenta de que sería mucho más cómodo enviar los datos de posición a una base de datos en tiempo real, y a través de una aplicación celular, mostrar los datos en un mapa que vaya actualizando la posición. Los módulos utilizados son los mismos, pero tiene una carga más importante de programación y desarrollo de software.

Para un funcionamiento más compacto, sería aconsejable utilizar el módulo SIM-808: GSM/GPRS + GPS, ya que soluciona el cableado de algunos módulos. La bluepill fue necesaria ya que necesitamos por lo menos 2 puertos de comunicación UART junto con la ventaja de tamaño, precio y capacidad de procesamiento, ya que un Arduino MEGA también dispone de ese número de puertos pero es mucho más grande y cara.

En la realidad, también podemos reemplazar los mensajes de consola por mensajes de texto enviados al módulo GPRS, o a través de una aplicación de celular.

Referencias

- GSM/GPRS:

  https://www.bbc.com/mundo/noticias-37247130

- Comandos AT:

  https://www.elecrow.com/wiki/images/2/20/SIM800_Series_AT_Command_Manual_V1.09.pdf

  https://www.estudioelectronica.com/wp-content/uploads/2018/09/ISTD-034.pdf

- Datasheet SIM800L:

  https://img.filipeflop.com/files/download/Datasheet_SIM800L.pdf

- GPS:

  https://www.geekfactory.mx/tutoriales/tutoriales-arduino/shield-o-modulo-gps-con-arduino/#:~:text=La%20tecnología%20GPS%20(Global%20Positioning,%2C%20Rumbo%2C%20Tiempo%2C%20etc.

- Datasheet GPS Gy-neo6 mv2:

  https://www.u-blox.com/sites/default/files/products/documents/NEO-6_DataSheet_(GPS.G6-HW-09005).pdf

- Manual de Referencia GPS Ublox:

  https://www.u-blox.com/sites/default/files/products/documents/u-blox6_ReceiverDescrProtSpec_%28GPS.G6-SW-10018%29_Public.pdf#page=30