Universidad Nacional de La Matanza

Técnicas Digitales II

Sistema de Acceso para Estacionamientos

Alumnos: Kelly Tomás, Lezcano Nicolaz, Olmedo Esteban

Comisión: 1949

Curso: 01

Días: Lunes 19:00-23:00 y Jueves 19:00-23:00

Profesores: Santos, Fidel Ignacio
Lucas, Guerrero

Año: 2020

Índice:

1. Introducción

I. Descripción

Lo que se busca en este proyecto es lograr la automatización del ingreso y egreso de vehículos a un estacionamiento, resolviendo los problemas generados por dichos vehículos y sus respectivos conductores. Este sistema es ideal para empresas, establecimientos educativos o incluso un simple estacionamiento. En este proyecto pusimos a prueba todos nuestros conocimientos adquiridos hasta ahora en electrónica y programación. Utilizaremos tecnología RFID para tener un mejor control de los usuarios. El sistema fue diseñado para un máximo de 8 automóviles basándonos en los lugares disponibles que nos ofrece la maqueta, pudiéndose aumentar la capacidad en el caso de que se trate de un estacionamiento más amplio.

II. Objetivos

- Optimizar y controlar el ingreso al estacionamiento de un establecimiento.
- Simplificar las tareas del personal encargado del estacionamiento por medio de la transmisión de los datos monitoreados hacia los dispositivos móviles de dichas personas.
- Generar el menor costo posible a la hora de implementarlo para garantizar el acceso a este sistema a la mayor cantidad de instituciones y sacarle el mayor de los provechos a los componentes, dándoles múltiples funcionalidades.
- Evitar el congestionamiento de tránsito en la entrada del estacionamiento que obstruya e interfiera en el tránsito normal de la avenida.

III. Alcance

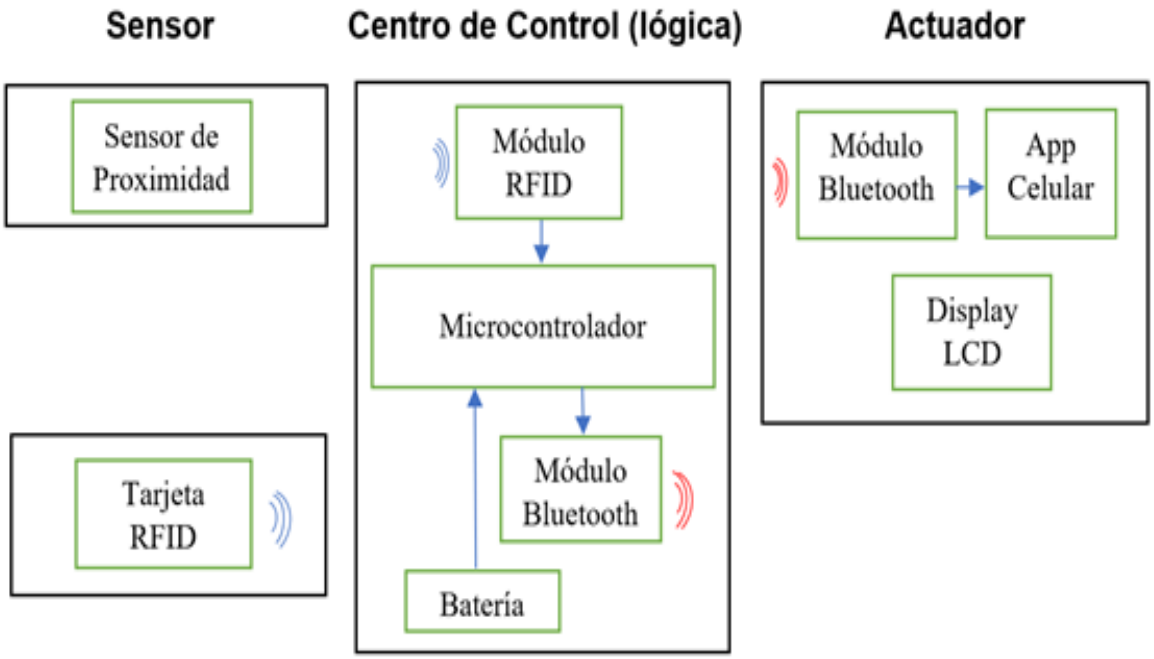
Nuestro sistema es una herramienta que busca optimizar y controlar el ingreso de vehículos a un estacionamiento, como podría ser el caso de la Universidad Nacional de La Matanza como así de otras instituciones y/o empresas.

El propósito del proyecto es restringir el ingreso a las personas no autorizadas, y aumentar la velocidad del flujo de vehículos, a través de la fácil visualización de lugares disponibles. Como resultado se busca, además, no alterar el tránsito de calles o avenidas en donde se encuentre la entrada al estacionamiento.

2. Funcionamiento

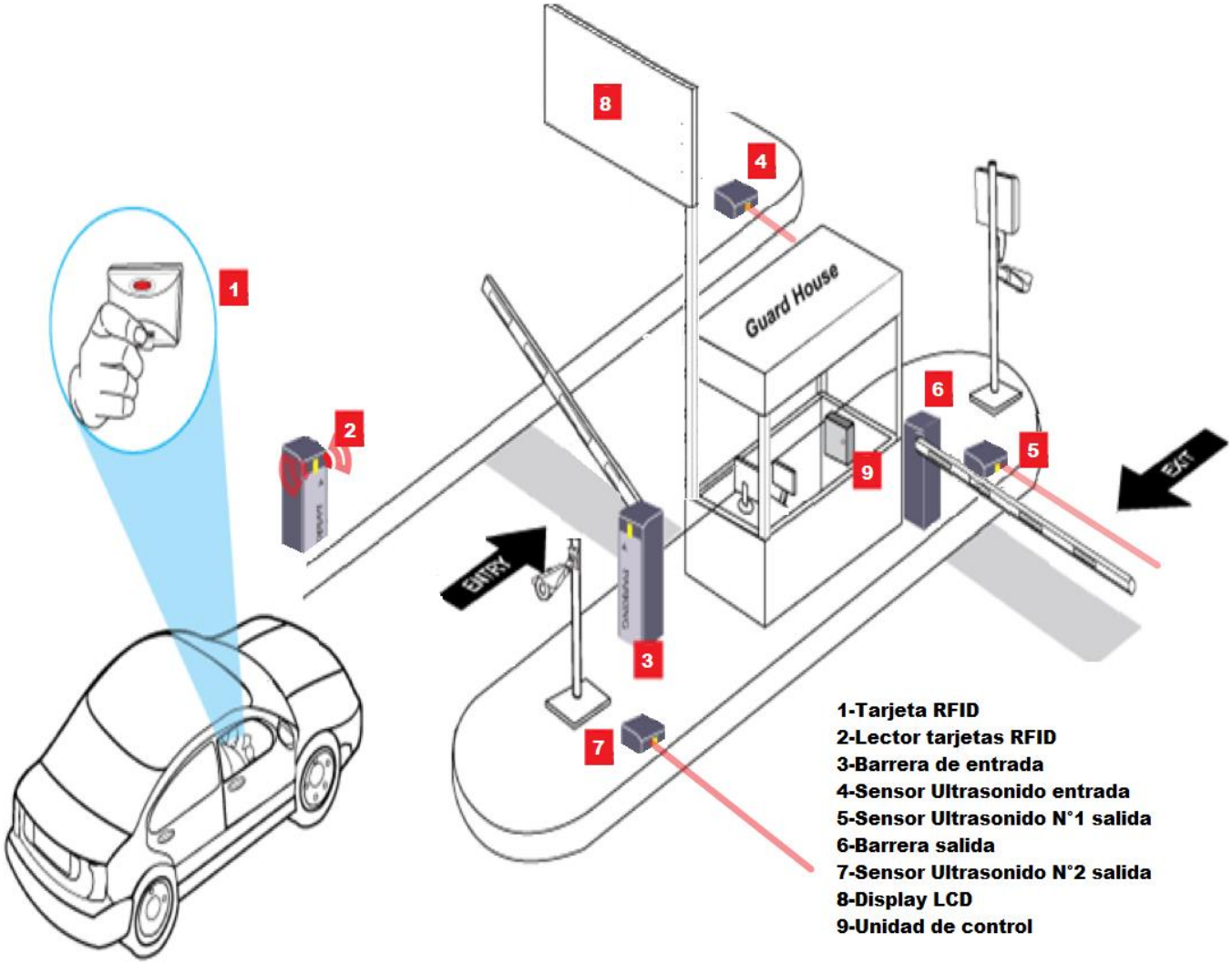
Por medio del Microcontrolador STM32F411 manejaremos un módulo RFID para identificar y autorizar o no, según corresponda, el ingreso del vehículo. Habrá un par de sensores que detecten cuando un automóvil ingrese y cuando uno sale, dicha información será procesada y mostrada en un display LCD para conocer la disponibilidad que hay en el estacionamiento, junto a una leyenda que indique si la persona está autorizada para ingresar.

Por otro lado, mediante un módulo Bluetooth se comunicará al celular del personal de seguridad el estado del estacionamiento. Si hay alguna eventualidad que requiera atención, la persona que está por ingresar podrá tocar un botón de emergencia que enviará una alerta al personal para que pueda acudir rápidamente a ayudar.



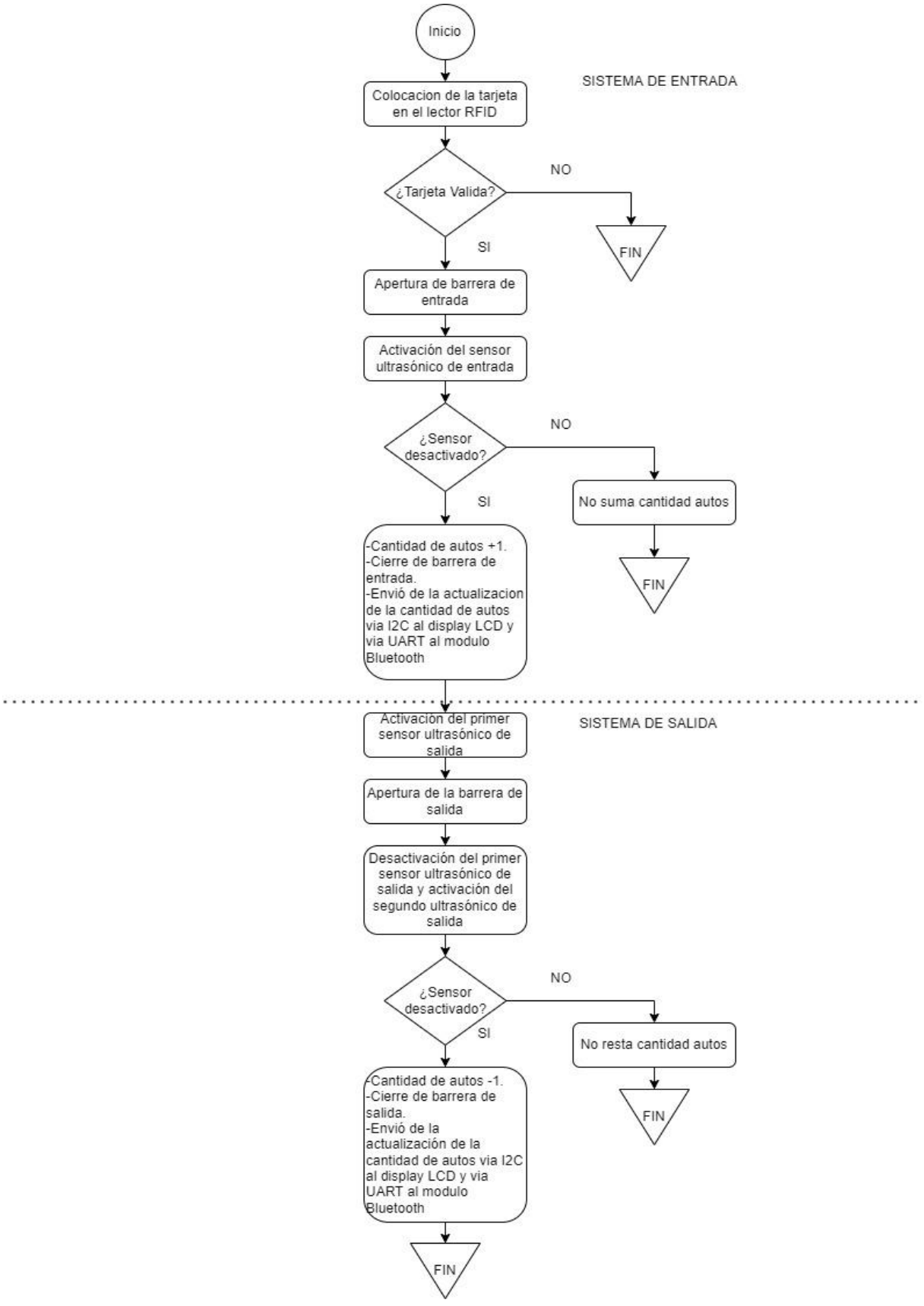
I. Proceso de funcionamiento

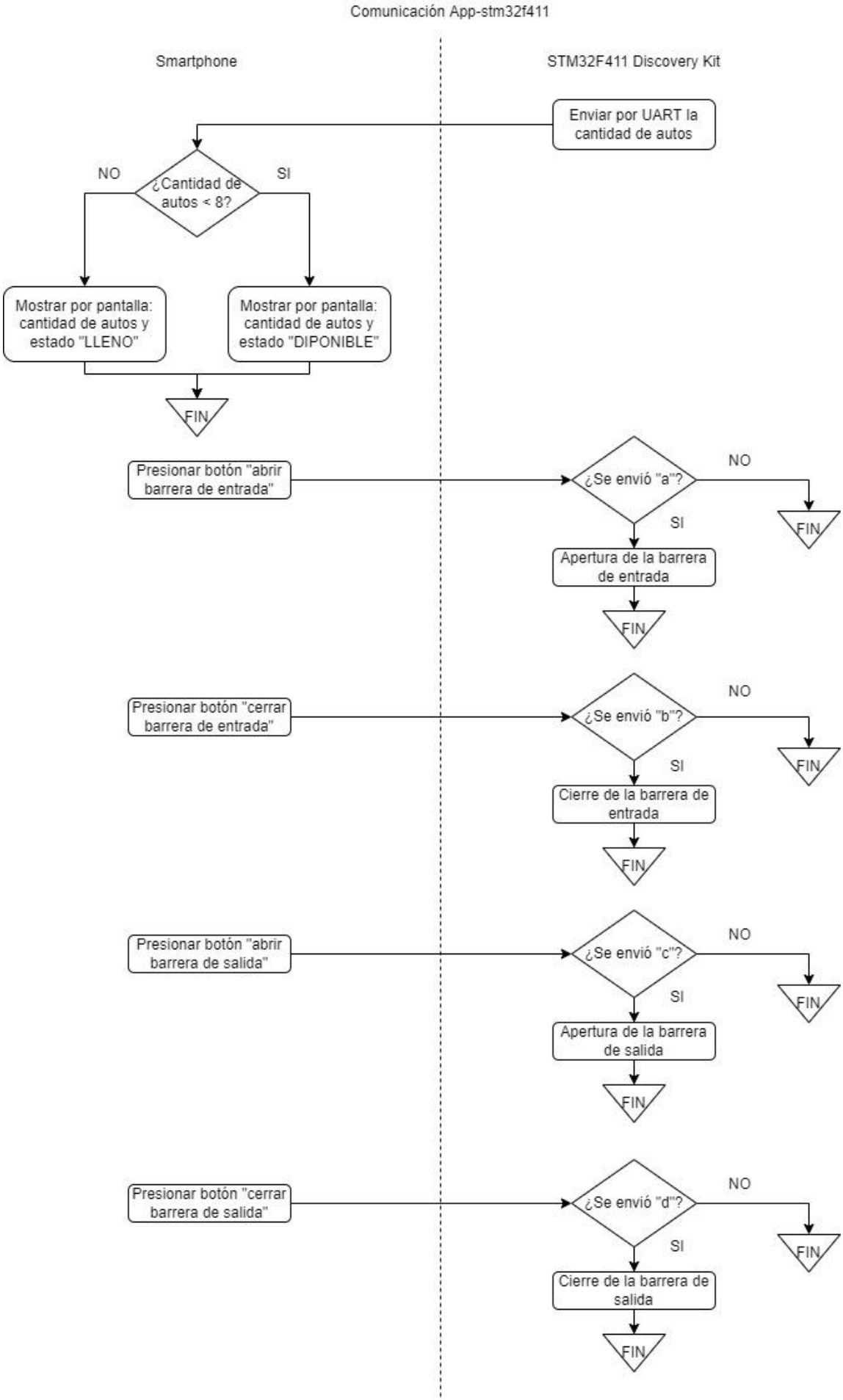
1. El usuario apoya su tarjeta identificadora en el módulo RFID que estará al principio de la entrada del estacionamiento.
2. Luego habrá dos condiciones, si hay lugar en el estacionamiento y si la tarjeta identificadora del usuario es válida. En caso de que ambas condiciones se cumplan la barrera se levanta. Dicha barrera se bajará una vez que el vehículo pase por completo. Esto lo vamos a corroborar con un sensor ultrasónico el cual va a estar colocado dos metros más delante de la barrera de entrada. Cuando este se activa es porque el vehículo está siendo detectado, una vez que el vehículo paso por completo el sensor se desactiva y automáticamente le comunica a la microcontroladora que se agregó un auto al estacionamiento.



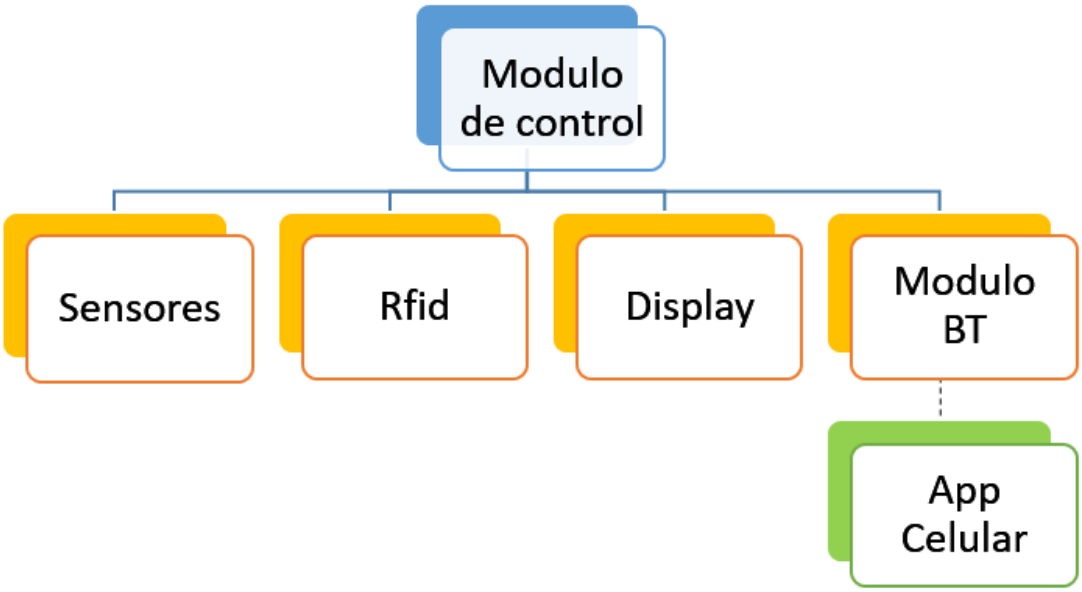
3. El proceso de salida cuenta con dos sensores ultrasónicos. El primero de estos lo que hace es levantar la barrera de la salida una vez que es activado por un vehículo. La barrera va a volver a bajarse una vez que el segundo sensor se desactiva, previamente a ser activado por el paso del vehículo. Inmediatamente se comunica con la microcontroladora para que reste un vehículo de los espacios ocupados.
4. Cada proceso que se realiza es recopilado por la microcontroladora. La cual se lo envía por medio de bluetooth al Smartphone del supervisor y por medio de I2C al LCD que se encuentra en la entrada del estacionamiento. Para así brindarle la información, no solo al supervisor, sino a las personas que están haciendo la fila para ingresar y no saben si hay o no lugares disponibles.

II. **Diagramas de estados:**




Comunicación App-stm32f411
Smartphone
STM32F411 Discovery Kit
Enviar por UART la cantidad de autos
¿Cantidad de autos < 8?
NO
SI
Mostrar por pantalla: cantidad de autos y estado "LLENO"
Mostrar por pantalla: cantidad de autos y estado "DIPONIBLE"
FIN
Presionar botón "abrir barrera de entrada"
¿Se envió "a"?
NO
SI
FIN
Apertura de la barrera de entrada
FIN
Presionar botón "cerrar barrera de entrada"
¿Se envió "b"?
NO
SI
FIN
Cierre de la barrera de entrada
FIN
Presionar botón "abrir barrera de salida"
¿Se envió "c"?
NO
SI
FIN
Apertura de la barrera de salida
FIN
Presionar botón "cerrar barrera de salida"
¿Se envió "d"?
NO
SI
FIN
Cierre de la barrera de salida
FIN

3. Diagrama de bloques



4. Software

III. Periféricos utilizados:

Las comunicaciones a efectuar entre los distintos módulos:

- La comunicación entre los sensores RFID RC522 y el microcontrolador será por medio de SPI.
- La comunicación entre los sensores HC-SR04 y el microcontrolador será manejada por el timer con los modos de operación correspondientes.
- La comunicación entre el display LCD y el microcontrolador será a través de I2C.
- La comunicación entre el módulo bluetooth (HC-05) y el celular del empleado será a través de UART.
- La comunicación entre el servo motor y el microcontrolador será a través de GPIO, utilizando PWM.

IV. Código

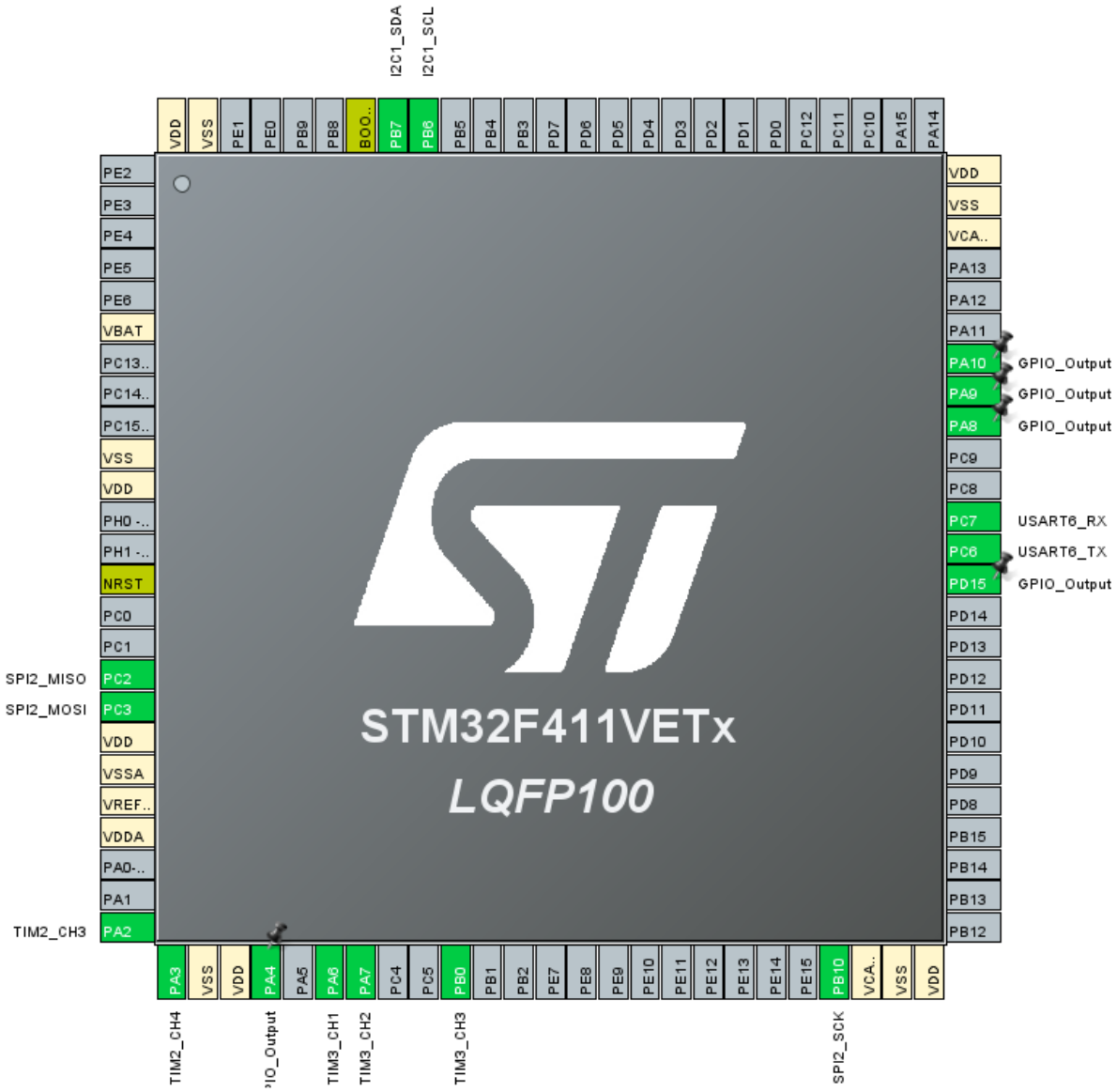
La interfaz que utilizamos para realizar el código y posteriormente debugearlo en la placa microcontroladora fue el STM32CubeIDE(v 1.6.1).

Enumeración de rutinas

En esta sección simplemente se van a enumerar todas las rutinas incluidas en el programa y la función que cumplen en el mismo en forma resumida.

- Rutina “main”: < Es la rutina principal y la que posee dentro todas las inicializaciones como el bucle principal en donde se encuentran todas las subrutinas >.
- Subrutina “HAL_Init”: < Inicialización y configuración de las funciones de la HAL (Hardware Abstraction Layer) >.
- Subrutina “SystemClock_Config”: < Configuración RCC >.
- Subrutina “MX_GPIO_Init”: < Inicialización GPIO >.
- Subrutina “MX_TIM1_Init”: < Inicialización del TIMER1 >.
- Subrutina “MX_TIM2_Init”: < Inicialización del TIMER2 >.
- Subrutina “MX_TIM3_Init”: < Inicialización del TIMER3 >.
- Subrutina “MX_I2C1_Init”: < Inicialización del I2C >.
- Subrutina “MX_USART6_UART_Init”: < Inicialización de la UART6 >.
- Subrutina “MX_SPI1_Init”: < Inicialización del SPI1 >.
- Subrutina “MFRC522_Init”: < Inicialización del lector RFID>.
- Subrutina “Display_Inicializacion”: < Inicialización del diplay LCD>.
- Subrutina “HAL_TIM_IC_CaptureCallback”: < Interrupción de captura TIM >.
- Subrutina “HAL_TIM_PeriodElapsedCallback”: <Interrupción de periodo TIM>.
- Subrutina “HAL_UART_RxCpltCallback”: < Interrupción de recepción UART >.
- Subrutina “HAL_UART_TxCpltCallback”: <Interrupción de transmisión UART>.

La etapa previa a la generación de código es lo que se conoce como la “selección de periféricos”. Esto nos da como resultado un archivo .ico, el cual, luego de realizar la asignación de todos los pines y periféricos a usar, nos permite, gracias el STM32CubeIDE, generar el código en el que se inicializan todos estos periféricos que seleccionamos. El resultado fue el siguiente:



Decidimos explicar y mostrar los fragmentos de código por categoría, con el fin de facilitar su entendimiento por parte del lector. Quedándonos de la siguiente manera:

- **While(1):** es el bucle en el que se ejecutan todas las funciones que engloban el funcionamiento final del proyecto. Dentro de este está cada función correspondiente a cada componente utilizado.

```
while (1){
        Sensores_Distancia();
        Lectura_RFID();
        Control_Barrera_Entrada();
        Control_Barrera_Salida();
        UART_Tx_Rx();
        Mensaje_Display();
}
```

- **Sensores_Distancia():** es la función correspondiente a los sensores ultrasónicos.

```
void Sensores_Distancia(void)
{
    if(Contador_Sensor_Distancia_Entrada>=925)
    {
            //Activo el Triggers del sensores ultrasonico
            HAL_GPIO_WritePin(TRIG_Entrada, 1);
            HAL_Delay(0.0001); //El delay tiene que ser de aproximadamente 100nseg
            //Desactivo el Trigger del sensor ultrasonico
            HAL_GPIO_WritePin(TRIG_Entrada, 0);

            __HAL_TIM_ENABLE_IT(ECHO_Timer, TIM_IT_CC1);
            Contador_Sensor_Distancia_Entrada=0;
    }

    if(Contador_Sensor_Distancia_Salida_1>=950)
    {
            HAL_GPIO_WritePin(TRIG_Salida_1, 1);
            HAL_Delay(0.0001); //El delay tiene que ser de aproximadamente 100nseg
            HAL_GPIO_WritePin(TRIG_Salida_1, 0);

            __HAL_TIM_ENABLE_IT(ECHO_Timer, TIM_IT_CC2);
            Contador_Sensor_Distancia_Salida_1=25;
    }

    if(Contador_Sensor_Distancia_Salida_2>=975)
    {
            HAL_GPIO_WritePin(TRIG_Salida_2, 1);
            HAL_Delay(0.0001); //El delay tiene que ser de aproximadamente 100nseg
            HAL_GPIO_WritePin(TRIG_Salida_2, 0);

            __HAL_TIM_ENABLE_IT(ECHO_Timer, TIM_IT_CC3);
            Contador_Sensor_Distancia_Salida_2=50;
    }
}
```

- **Lectura_RFID():** es la función correspondiente al módulo RFID RC522.

```
void Lectura_RFID(void) //3,3v=RST=3v, GND=0v, MISO: PC2, MOSI: PC3, SCK: PB10, SDA: PA4
{
    if(Contador_Lector_RFID>=999) //para que lo realice cada un segundo aproximadamente
    {
            Estado=MFRC522_Request(PICC_REQIDL, String);
            Estado=MFRC522_Anticoll(String);
            memcpy(Numero_Serie, String, 5); //Se alamacena el numero de serie de la tarjeta RFID
```

```
            if(Estado==MI_OK)
            {
                    if((Numero_Serie[0]==50) && (Numero_Serie[1]==132) && (Numero_Serie[2]==177)
&& (Numero_Serie[3]==1) && (Numero_Serie[4]==6)) //Tarjeta 1
                    {
                            __HAL_TIM_SET_COMPARE(Servo_Timer, Servo_Entrada, 2); //Abro la
barrera. Con un CCR de 2 el servo esta a 90°, ciclo de actividad de 2ms
                            Flag_RFID=ON;
                    }

                    else if((Numero_Serie[0]==34) && (Numero_Serie[1]==209) &&
(Numero_Serie[2]==153) && (Numero_Serie[3]==1) && (Numero_Serie[4]==107)) //Trajeta 2
                    {
                            __HAL_TIM_SET_COMPARE(Servo_Timer, Servo_Entrada, 2); //Abro la
barrera. Con un CCR de 2 el servo esta a 90°, ciclo de actividad de 2ms
                            Flag_RFID=ON;
                    }

                    else if((Numero_Serie[0]==50) && (Numero_Serie[1]==80) &&
(Numero_Serie[2]==255) && (Numero_Serie[3]==1) && (Numero_Serie[4]==156))
                    {
                            __HAL_TIM_SET_COMPARE(Servo_Timer, Servo_Entrada, 2); //Abro la
barrera. Con un CCR de 2 el servo esta a 90°, ciclo de actividad de 2ms
                            Flag_RFID=ON;
                    }

                    else //if((Numero_Serie[1]!=132) && (Numero_Serie[1]!=209) &&
(Numero_Serie[1]!=80))
                    {
                            Flag_RFID=OFF;
                    }
            }
            Contador_Lector_RFID=0;
    }
}
```

- **Control_Barrera_Entrada():**

```
void Control_Barrera_Entrada(void) //Trabajan el Canal 1 del Timer 3 y el Lector RFID
{
    if(Flag_Control_Barrera_Entrada==1)
    {
            static uint8_t Estado_Ultrasonido_Entrada=IDLE; //Estado inicial

            switch(Estado_Ultrasonido_Entrada)
            {
                    case IDLE:
                            if(Distancia_cm_Entrada<5 && Distancia_cm_Entrada>0 && Flag_RFID==ON
&& Cantidad_Autos<8)
                            {
                                    Estado_Ultrasonido_Entrada=CONFIRM;
                            }
                    break;

                    case CONFIRM:
                            if(Distancia_cm_Entrada>5)
                            {
                                    Estado_Ultrasonido_Entrada=IDLE;
                                    __HAL_TIM_SET_COMPARE(Servo_Timer, Servo_Entrada, 1); //Cierro
la barrera. Con un CCR de 1 el servo esta a 0°, ciclo de actividad de 1ms
                                    Flag_Cruce_Entrada=ON;
                            }
                    break;
            }
            if(Flag_Cruce_Entrada==ON)
            {
                    Flag_Cruce_Entrada=OFF;
                    Flag_RFID=OFF;
                    Cantidad_Autos++; //Cuento autos
            }
            Flag_Control_Barrera_Entrada=0;
    }
}
```

- **Control_Barrera_Salida():**

```
void Control_Barrera_Salida(void)
{
    //El PWM para los servos esta configurado para que al 100% de ciclo de actividad (19) sea un estado alto de 20ms
    /*__HAL_TIM_SET_COMPARE(Servo_Timer, Servo_Salida, 1); //Con un CCR de 1 el servo esta a 0°, ciclo de actividad de 1ms
    HAL_Delay(3000);
    __HAL_TIM_SET_COMPARE(Servo_Timer, Servo_Salida, 2); //Con un CCR de 2 el servo esta a 90°, ciclo de actividad de 2ms
    HAL_Delay(3000);*/

//----------Barrera de Salida--------->Trabajan el Canal 2 y el Canal 3 del Timer 3

    if(Flag_Control_Barrera_Salida==1)
    {
            static uint8_t Estado_Ultrasonido_Salida_1=IDLE; //Estado inicial

            switch(Estado_Ultrasonido_Salida_1)
            {
                    case IDLE:
                            if(Distancia_cm_Salida_1<5 && Distancia_cm_Salida_1>0 &&
Cantidad_Autos>0)
                            {
                                    Estado_Ultrasonido_Salida_1=CONFIRM;
                                    __HAL_TIM_SET_COMPARE(Servo_Timer, Servo_Salida, 2); //Abro la barrera. Con un CCR de 2 el servo esta a 90°, ciclo de actividad de 2ms
                            }
                    break;

                    case CONFIRM:
                            if(Distancia_cm_Salida_1>5)
                            {
                                    Estado_Ultrasonido_Salida_1=IDLE;
                                    Flag_Cruce_Salida_1=ON;
                            }
                    break;
            }

            static uint8_t Estado_Ultrasonido_Salida_2=IDLE; //Estado inicial

            switch(Estado_Ultrasonido_Salida_2)
            {
                    case IDLE:
                            if(Distancia_cm_Salida_2<5 && Distancia_cm_Salida_2>0 &&
Flag_Cruce_Salida_1==ON && Cantidad_Autos>0)
                            {
                                    Estado_Ultrasonido_Salida_2=CONFIRM;
                            }
                    break;

                    case CONFIRM:
                            if(Distancia_cm_Salida_2>5)
                            {
                                    Estado_Ultrasonido_Salida_2=IDLE;
                                    __HAL_TIM_SET_COMPARE(Servo_Timer, Servo_Salida, 1); //Cierro la barrera. Con un CCR de 1 el servo esta a 0°, ciclo de actividad de 1ms
                                    Flag_Cruce_Salida_2=ON;
                            }
                            break;
            }

            if(Flag_Cruce_Salida_2==ON)
            {
                    Flag_Cruce_Salida_1=OFF;
                    Flag_Cruce_Salida_2=OFF;
                    Cantidad_Autos--; //Descuento autos
            }

            Flag_Control_Barrera_Salida=0;
    }
}
```

- **UART_Tx_Rx():** es la función que se encarga de transmitir y recibir los datos entre la microcontroladora y la app móvil.

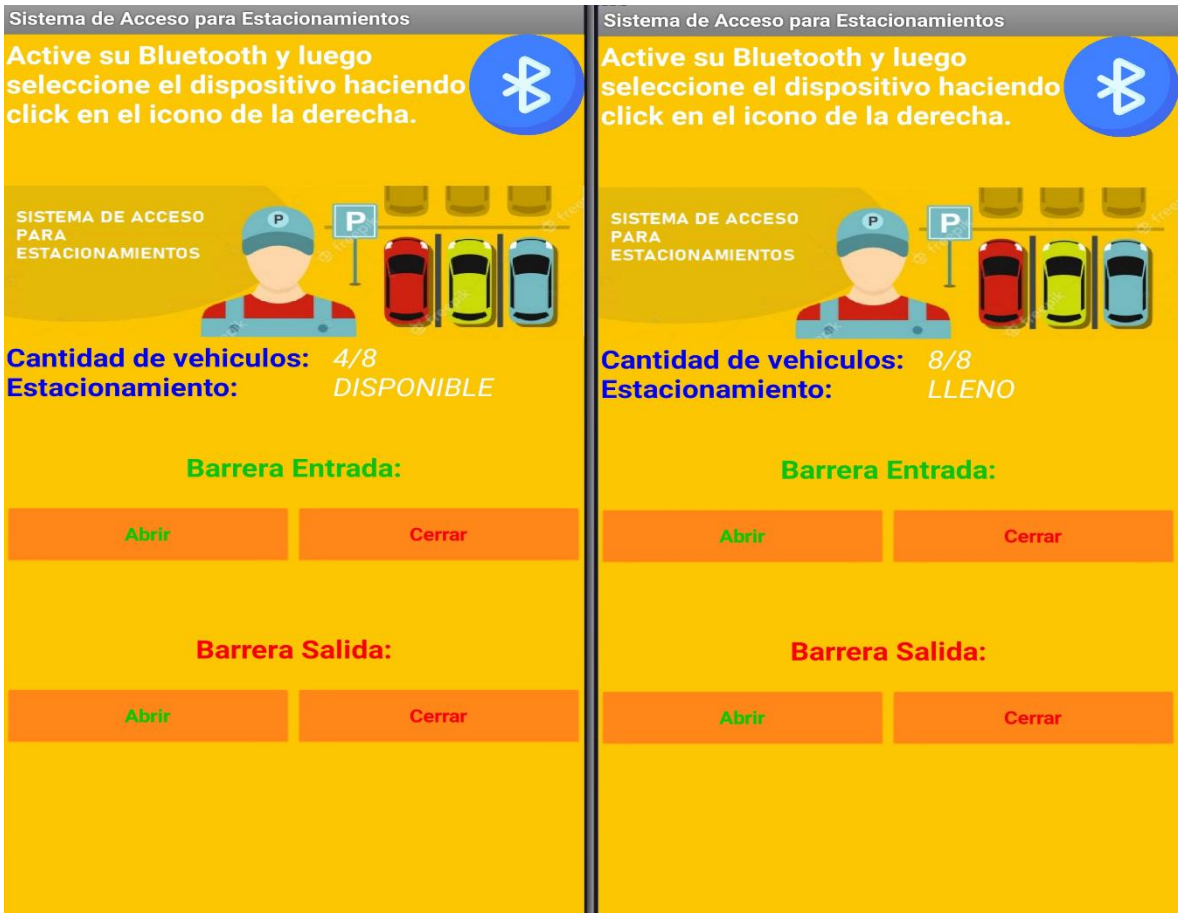
```
void UART_Tx_Rx(void)
{
    if(Contador_UART>=999)
    {
            if(Tx_Estado==LISTO) //Transmito
            {
                    switch(Cantidad_Autos)
                    {
                            case 8:
                                    sprintf(Tx_Buffer_Uart, " %d/8 \n LLENO \n", Cantidad_Autos);
                                    HAL_UART_Transmit_IT(&huart6, (uint8_t *)Tx_Buffer_Uart,
strlen(Tx_Buffer_Uart));
                                    break;

                            default:
                                    sprintf(Tx_Buffer_Uart, " %d/8 \n DISPONIBLE \n",
Cantidad_Autos);
                                    HAL_UART_Transmit_IT(&huart6, (uint8_t *)Tx_Buffer_Uart,
strlen(Tx_Buffer_Uart));
                                    break;
                    }

                    Tx_Estado==OCUPADO;
            }

    Contador_UART=0;
    }

    if(Rx_Dato==LLENO) //Recibo
    {
            switch(Rx_Buffer_Uart[0])
            {
                    case 'a': //Abrir la barrera de entrada
                            __HAL_TIM_SET_COMPARE(Servo_Timer, Servo_Entrada, 2); //Abro la
barrera. Con un CCR de 2 el servo esta a 90°, ciclo de actividad de 2ms
                    break;

                    case 'b': //Cerrar la barrera de entrada
                            __HAL_TIM_SET_COMPARE(Servo_Timer, Servo_Entrada, 1); //Cierro la
barrera. Con un CCR de 1 el servo esta a 0°, ciclo de actividad de 1ms
                    break;

                    case 'c': //Abrir la barrera de salida
                            __HAL_TIM_SET_COMPARE(Servo_Timer, Servo_Salida, 2); //Abro la
barrera. Con un CCR de 2 el servo esta a 90°, ciclo de actividad de 2ms
                    break;

                    case 'd': //Cerrar la barrera de salida
                            __HAL_TIM_SET_COMPARE(Servo_Timer, Servo_Salida, 1); //Cierro la
barrera. Con un CCR de 1 el servo esta a 0°, ciclo de actividad de 1ms
                    break;

                    default:
                    break;
            }

            //Rx_Buffer_Uart[0]="";
            Rx_Dato=VACIO;
    }
}
```

- **Mensaje_Display():** es la función que se encarga de transmitir los datos desde la microcontroladora y hacia el display.

```
void Mensaje_Display(void){
        if(Contador_Display>=999){

                Estacionamientos_Disponibles=8-Cantidad_Autos;

                Display_Ubicar_Cursor(1, 13);
                sprintf(Estacionamientos, "%d", Estacionamientos_Disponibles);
                Display_Enviar_String(Estacionamientos);
                //Display_Enviar_String(" ");

                Contador_Display=0;
                HAL_GPIO_TogglePin(LED_Azul);
        }
}
```

V. Interfaz

La interfaz que va a utilizar desde el celular el encargado/ supervisor fue realizada con el programa appInventor. Está disponible para dispositivos Android descargándola desde el siguiente link: SAE.apk. La aplicación cuenta con la información de la cantidad de autos que hay en el estacionamiento, si éste está lleno o con lugares disponibles y trae las opciones de subir o bajar ambas barreras.



5. Hardware

VI. Componentes

Microcontrolador (STM32F411E-DISCO Discovery Kit): es el cerebro del sistema, hacia él llega la información proveniente de distintos tipos de sensores y se encarga de procesar esos datos para generar una respuesta acorde al problema que debemos resolver, esta respuesta se ve reflejada en distintos dispositivos de salida. Para grabar las órdenes que luego ejecutará el microcontrolador, programa lenguaje C/C++. El IDE que utilizamos es el que nos proporciona la marca de la microcontroladora STM32CubeIDE(v. 1.6.1). Decidimos utilizar esta microcontroladora por sobre la LPC1769 debido a su gran alcance hoy en día y a la gran cantidad de periféricos que nos otorga.

Características:

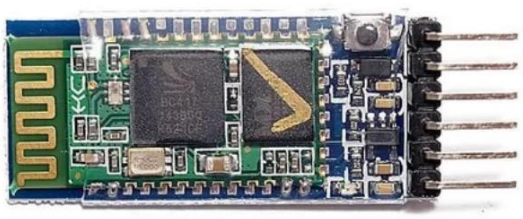
- Microcontrolador STM32F411VET6, 512KB de memoria flash, 128KB de RAM en 100 pines LQFP.
- ST-LINK/V2 integrado con selector de modo de selección para usar el kit como un ST-LINK/V2 independiente.
- Diseñado para ser alimentado por USB o un suministro externo de 5V.
- Puede suministrar aplicaciones con 3V y 5V.
- L3GD20: Giroscopio de salida digital de tres ejes y sensor de movimiento ST MEMS
- LSM303DLHC: Sistema en paquete ST MEMS que incluye un sensor de aceleración lineal digital 3D y un sensor magnético digital en 3D.

- MP45DT02: Sensor de audio ST MEMS, micrófono digital omnidireccional.
- CS43L22: audio DAC con controlador de altavoz clase D integrado.
- Cuatro LED de usuario.
- Dos botos pulsadores (usuario y reset).
- USB OTG con conector micro-AB (STMicroelectronics, 2017).

Lector y tarjetas RFID (RFID-RC522): es un sistema de identificación a través de tarjetas/tags, que se utiliza como una llave electrónica permitiendo o no el acceso hacia área restringidas. El mismo trabaja a una frecuencia de 13,56Mhz con etiquetas de radio frecuencia pasivas, por lo que es necesario que la tarjeta se acerque al lector de tarjetas ya que es de corto alcance.

En nuestro caso actúa como una entrada para nuestra unidad de control. Si se autoriza el ingreso se le suma uno a la cantidad de espacios ocupados por los vehículos.

Módulo Bluetooth (HC-05): permite una conexión inalámbrica entre dispositivos alejados, lo cual facilita en términos de practicidad la comunicación. Además, les permite a los operadores una mayor movilidad por el establecimiento. La limitante es una distancia máxima de aproximadamente 10 metros.



Display LCD 16x2 (QAPASS 1602A): pantalla que utilizamos como interfaz para informar al conductor acerca de la disponibilidad o no de un espacio para estacionar. Lo utilizaremos como representación de un cartel luminoso de grandes dimensiones que estará situado arriba de la entrada del estacionamiento, para que aquellos que estén haciendo la fila para poder ingresar sepan de antemano si hay o no lugar disponible.

Sensor de Proximidad (HCSR-04): mide la distancia entre el sensor y un objeto calculando el tiempo que tarda la onda de ultrasonido en salir del emisor rebotar con el objeto e incidir en el receptor. Cumple la función de detectar cuando sale un auto para levantar la barrera y restar uno a la cantidad de autos estacionados (sumamos cuando se autoriza el ingreso y levanta la barrera).

Servo Motor(SG90): vamos a implementar la utilización de un servomotor (SG90) para simular las barreras de nuestro estacionamiento. Vamos a contar con dos, uno para la entrada y otro para la salida.

Alimentación: El módulo que abarca el microcontrolador, el display LCD, los sensores, los motores y el módulo bluetooth estarán conectados a una fuente de 5V 10A conectada a la red de distribución. Se tomó en cuenta que todos los componentes (contando sensores, display LCD, módulo bluetooth, placa controladora y motores) operan en 5V.

VII. Circuito Impreso

El circuito fue realizado utilizando el programa Kicad. Fue diseñado de forma tal que la microcontroladora pueda ser reutilizable para otros usos independientes al proyecto, ó en caso de que esta cuente con una falla solo haya que reemplazarla por una que funcione correctamente, sin tener que cambiar componentes.

Pinout adoptado:

HC-05:	PC6-Tx PC7-Rx
HC-SR04:	PA8-Trigger PA6-Echo
HC-SR04:	PA9-Trigger PA7-Echo
HC-SR04:	PA10-Trigger PB0-Echo

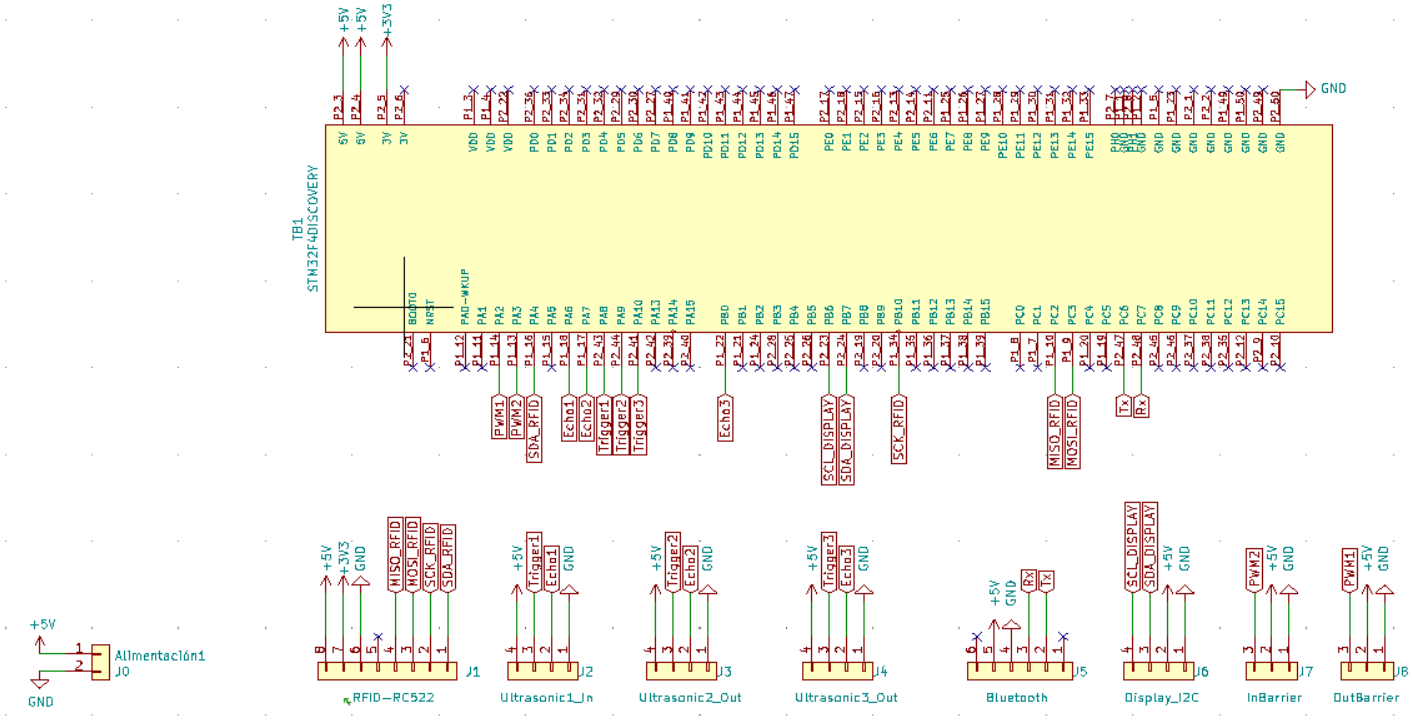
LCD 16x2: PB6-SCL
PB7-SDA

Servo Motor: PA2-PWM

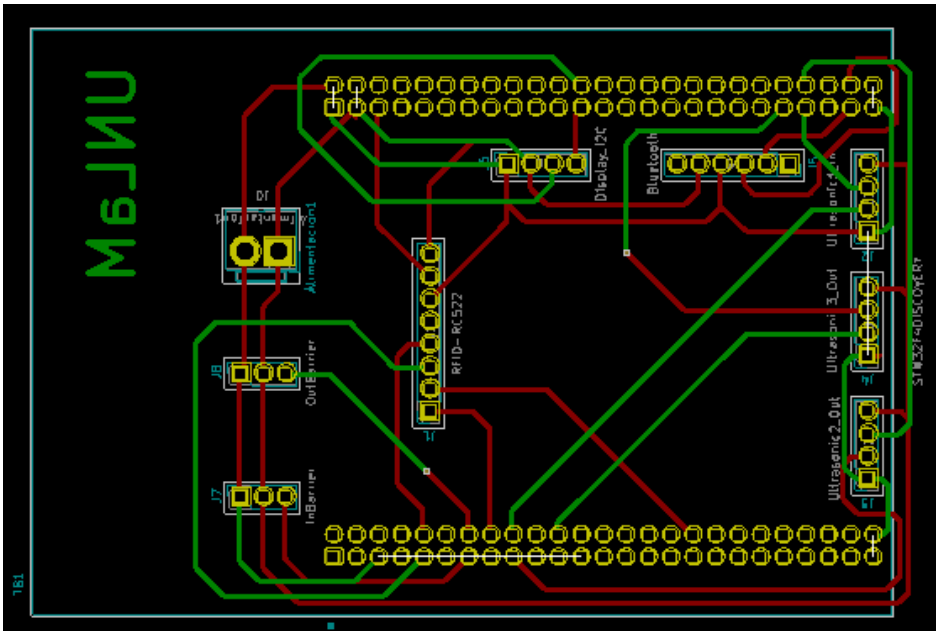
Servo Motor: PA3-PWM

RC522: PA4-SDA
PB10-SCK
PC3-MOSI
PC2-MISO

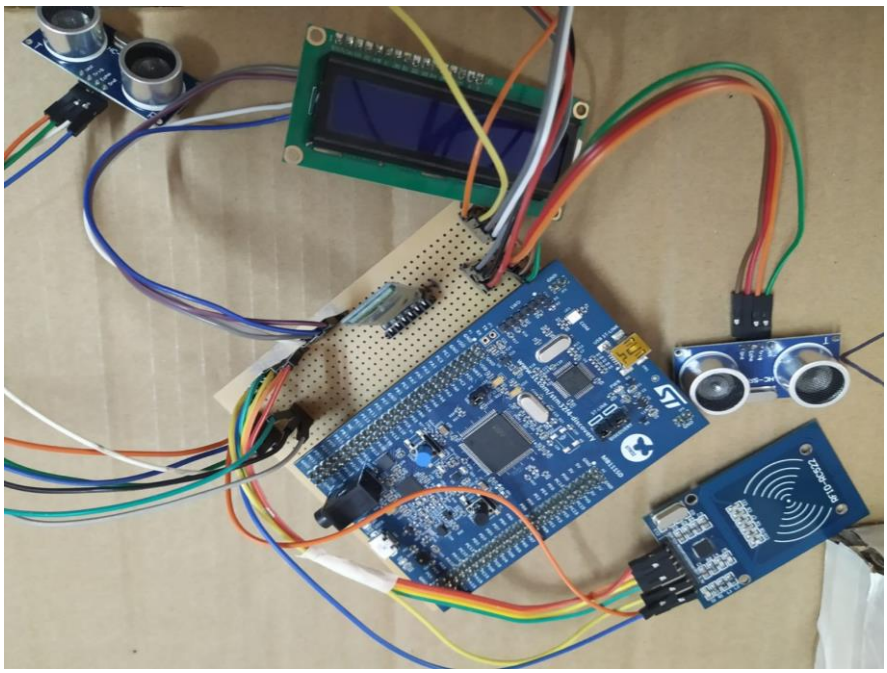
Esquemático:



PCB:



Como bien se dijo, realizamos el diseño del circuito mediante el programa Kicad y posteriormente imprimimos dicho circuito en una plaqueta de pertinax doblefaz.



VIII. **Maqueta**

La maqueta fue desarrollada reutilizando un estacionamiento de juguete al cual le realizamos las respectivas modificaciones para que quede de la siguiente forma. De esta manera ahorramos en el costo de tener que comprar materiales y, además, le dimos una “segunda vida” a este juguete.

Ilustración 1:Maqueta sin las modificaciones hechas.

Ilustración 2: Maqueta sin las modificaciones hechas

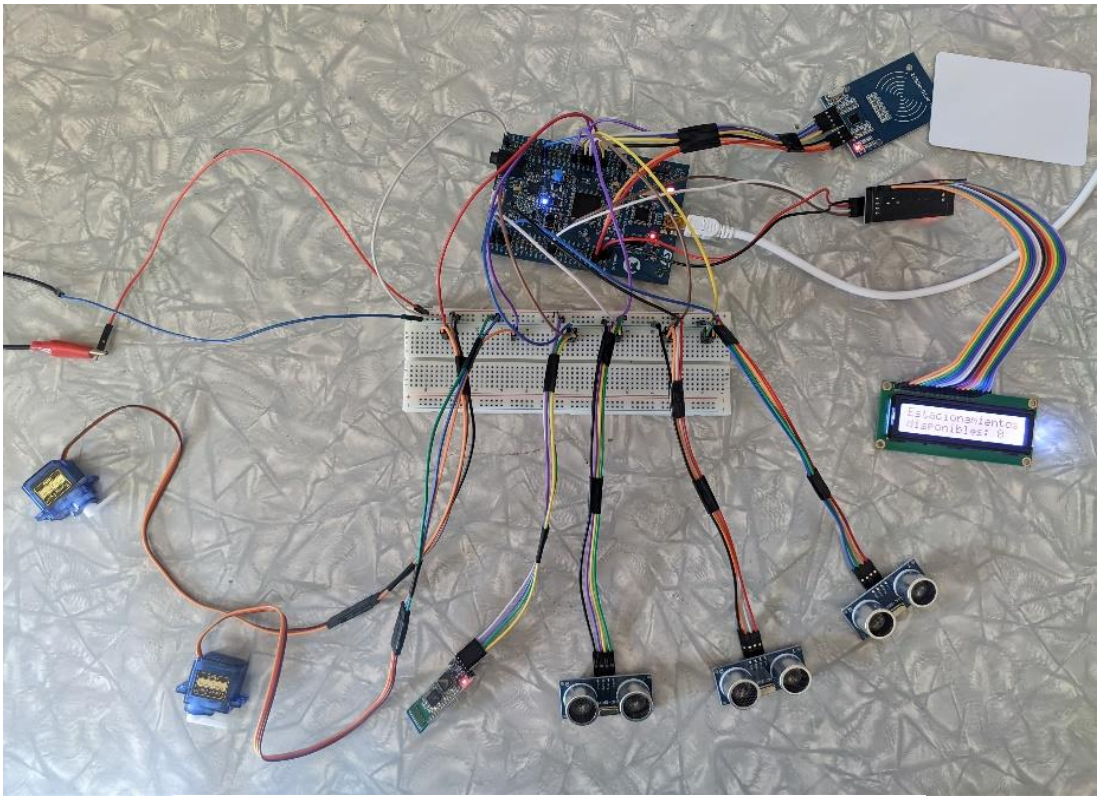


Ilustración 3:Prototipo

Ilustración 4: LCD en funcionamiento

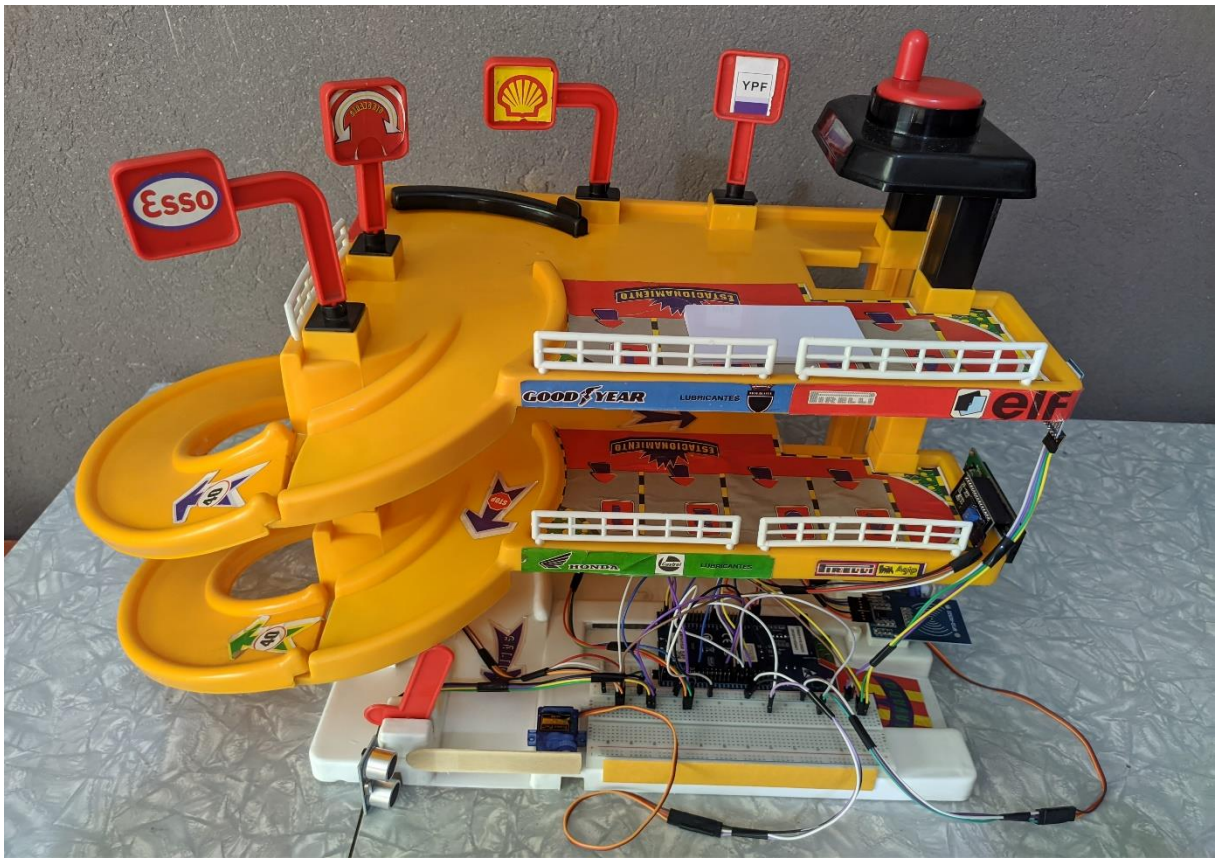


Ilustración 5: Maqueta Finalizada

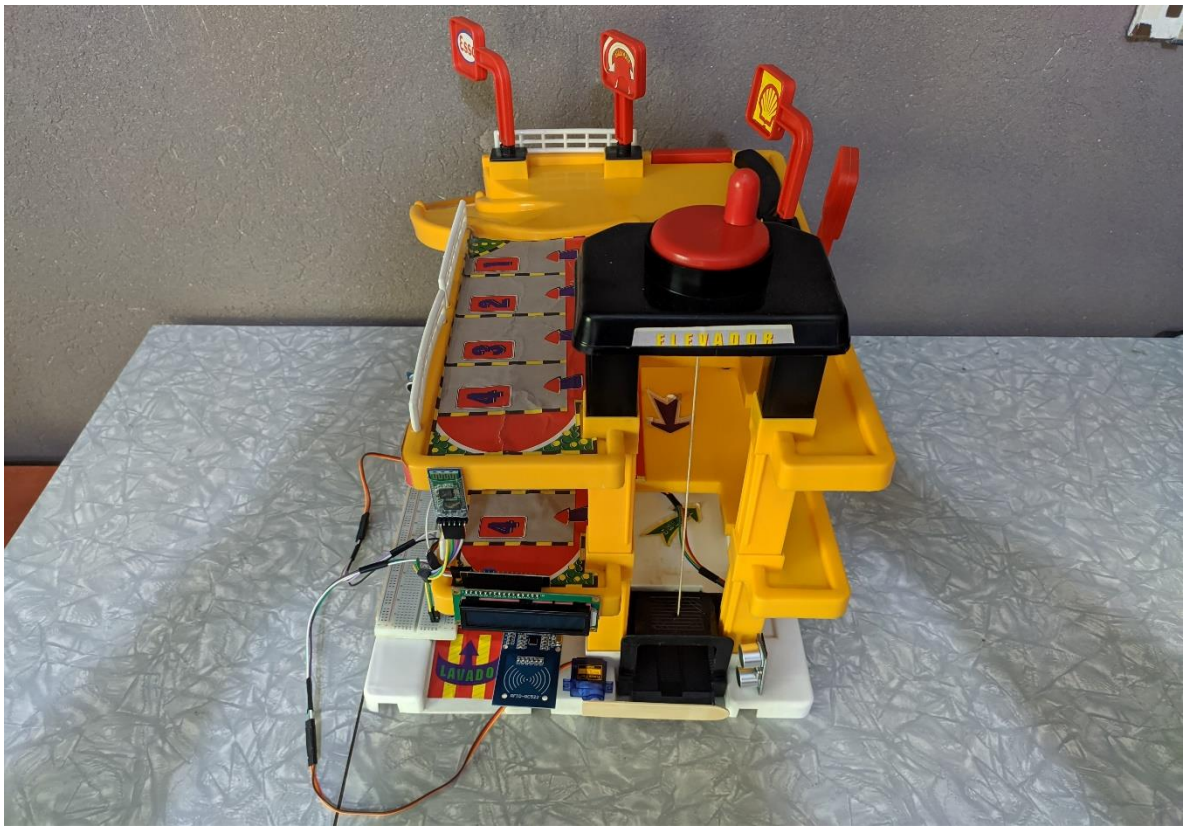


Ilustración 6: Maqueta Finalizada

6. Desarrollo

IX. Medidores de Distancia Ultrasónicos:

En nuestro proyecto utilizamos tres sensores ultrasónicos HC-SR04. Para el subsistema que se encarga de la salida se utilizaron dos de estos tres. Mientras que el restante lo utilizamos en el subsistema que se encarga de la entrada, cuya función es la de detectar si el auto ingreso en su totalidad al estacionamiento y, además, cerrar la barrera.

Los periféricos que utilizamos para poder controlar dichos componentes son mediante la configuración adecuada de tres timers, uno por cada ultrasónico.

Como su nombre lo indica, los sensores ultrasónicos miden distancia mediante el uso de ondas ultrasónicas. El cabezal emite una onda ultrasónica y recibe la onda reflejada que retorna desde el objeto. Los sensores ultrasónicos miden la distancia al objeto contando el tiempo entre la emisión y la recepción. Dicha distancia nos la devuelve en una variable la cual, mediante una fórmula matemática que nos provee el fabricante en la hoja de datos, la convertimos a una distancia en centímetros. De esta manera podemos saber la distancia a la cual está el objeto de nuestro sensor. Y de esta forma podemos saber si detecto o no un auto en el caso de que este último haya pasado a una distancia menor a, por ejemplo, un metro.

X. Comunicación Bluetooth:

El proyecto consta de comunicación bluetooth para poder realizar la recepción y transmisión de datos con la app del Smartphone.

Para realizar dicha comunicación se utilizó un módulo de bluetooth HC-05 que cumple con las especificaciones del estándar Bluetooth 2.0 a 2.4 GHz que es perfectamente compatible con celulares o Smartphone Android. El mismo fue configurado por UART. Además de utilizar la UART para la configuración de dicho componente, se utilizó también un Timer de propósito general con el fin de ser utilizado para sincronizar los datos enviados al celular.

Los módulos Bluetooth se pueden comportar como esclavo o maestro, los cuales sirven para escuchar peticiones de conexión y otros para generar peticiones de conexión. Es por este motivo que para poder utilizar el HC-05 como esclavo en la microcontroladora STM32F411 previamente tuvimos que configurar el módulo bluetooth en un Arduino UNO por parámetros AT para que el mismo opere como esclavo y de esta forma poder realizar la comunicación deseada con la aplicación móvil.

XI. **Lector de tarjetas RFID:**

Nuestro proyecto incluye la utilización de un lector de tarjetas RFID, el cual es el primer “eslabón” que entra en funcionamiento en lo que engloba a la totalidad del sistema de estacionamiento en sí.

Es un módulo que permite tanto leer como escribir etiquetas RFID utilizando la tecnología MIFARE. Este mismo utiliza un sistema de modulación y demodulación de 13.56MHz, frecuencia que en la actualidad utiliza la tecnología RFID. El RC522 se comunica por SPI, por lo que se puede implementar con cualquier microcontrolador con interfaz SPI, como por ejemplo la stm32f411.

Para el desarrollo del código debimos utilizar la librería correspondiente al RC522 y, además, su archivo en formato “.c” para poder inicializarlo.

Para leer la información codificada en una tarjeta RFID pasiva, se coloca cerca del lector que genera un campo electromagnético que hace que los electrones se muevan a través de la antena de la etiqueta y posteriormente alimenten el chip. Cuando esto sucede, el chip alimentado es capaz de enviar la información almacenada en la etiqueta RFID a través de la radiofrecuencia al stm32f411.

XII. **Display 16x2:**

En un comienzo se iba a trabajar con el display mediante GPIO Output, es decir, enviar unos y ceros según corresponda para generar las correspondientes instrucciones, pero con este método se necesitaban utilizar 8 pines de datos, RS, RW y E. Ante este problema se optó por utilizar un módulo I2C para el display 16x2 y gracias a este cambio solamente se utilizan dos pines, SDA y SCL. Gracias a la introducción

del protocolo I2C y a la utilización de una librería en vez de tener que enviar unos o ceros a 11 pines del display basta con enviar la instrucción en hexadecimal para realizar la función de inicialización del LCD. Otra de las ventajas que proporciona dicha librería es que permite enviar de strings.

XIII. **Barrera de servomotores:**

Para la realización de las barreras de entrada y salida se optó por la utilización de servomotores, ya que en función de la información que se le ingresa al pin de PWM se puede controlar los grados de rotación del mismo, desde 0° a 180°. Para el caso específico de unas barreras solamente es necesario mover el servomotor de 0° a 90° o viceversa, según corresponda. Los servomotores en la entrada de PWM necesitan una señal con una amplitud de 5v y un período de 20ms, al generar un ciclo de actividad de 1ms se obtiene 0°, al generar uno de 1,5ms se obtienen 90° y al generar uno de 2ms se obtienen los 180°. Lo anteriormente dicho corresponde a un valor lógico alto de 5v para el PWM, pero el microcontrolador STM32F411 genera como máximo una amplitud de 3v cuando se configura el canal de un timer como generador de PWM. En este caso no presentan mayor inconveniente los 3v de amplitud máxima que ingresan al pin de PWM porque los servomotores se encuentran alimentados por los 5v que indica el fabricante, lo que cambia es que para obtener 0° se debe generar un ciclo de actividad que dure 1ms y para 90°, se utiliza un ciclo de actividad de 2ms.

7. Conclusiones

Si bien se trata de un proyecto que ya existía de antaño, pudimos demostrar que se puede realizar un sistema tan complejo como es este gastando poco dinero.

Lo que se puede destacar del proyecto es que pudimos poner en funcionamiento la mayoría de los periféricos que nos brinda la stm32f411(UART,I2C,SPI,Timmer,GPIO,PWM). De esta forma demostramos nuestros conocimientos sobre lo visto en la materia.

XIV. **Análisis de Mercado:**

A la hora de realizar un análisis de mercado sobre nuestro proyecto llegamos a las siguientes conclusiones:

- En caso de ser un producto personalizado para un único cliente, es correcta la idea de utilizar una microcontroladora stm32f411, debido a que si el cliente nos pide agregarle más funciones al sistema la microcontroladora nos lo permite debido a su amplia variedad de periféricos para usar. Las ventajas serían entonces la flexibilidad para agregarle funcionalidades extras demandadas por el cliente. Mientras que la desventaja sería el precio.

- En caso de ser un producto vendido para una cierta cantidad de clientes (10-20 clientes), es conveniente usar la microcontroladora stm32f103 (Blue Pill). La ventaja es su bajo costo, respecto a las stm32f411. Mientras que la desventaja sería que estaríamos más condicionados a la hora de querer agregarle funcionalidades extras debido a su limitada cantidad de periféricos.

- En caso de ser un producto vendido de forma masiva (más de 30 clientes), es conveniente diseñar una plaqueta PCB donde este el microprocesador stm32 Cortex®-M4 y todos los componentes electrónicos necesarios para cubrir las funcionalidades del producto en sí. La ventaja sería el bajo costo del producto respecto a usar una blue pill. Mientras que la desventaja es que no se puede agregar demasiadas funcionalidades extras a la misma, ya que en la mayoría de los casos eso significaría rediseñar un nuevo PCB y agregarle los componentes extras.

8. Referencias

- STM32F411-DATASHEET
- HC-05-DATASHEET
- LCD1602-DATASHEET
- HC-SR04-DATASHEET
- RC522-DATASHEET
- USERMANUALSTM32F4

9. Repositorio

Todo el contenido del proyecto, ya sea código, imágenes, pruebas, datasheets y hasta este mismo informe se puede encontrar en el siguiente repositorio de GitHub: click aquí.