Título:	Simulación Cochera Inteligente		
Ciclo Lectivo:	2016	**Curso:**	R2002

Integrantes	**Apellido Nombres**	**Legajo**	**Calificación individual**	**Fecha**
Kinast Alan		151.390-4		16/12/2016
Trautman Marcos		150.489-7		16/12/2016

Grupo	1

Profesor:	Ing. Trujillo Marcelo
Auxiliar/es Docente/s:	Ing. Hernán Goin

Observaciones primera entrega	
Observaciones segunda entrega	

Índice

1) Objetivos – pág 3
2) Temas Involucrados en Hardware de Cátedra – pág 4
3) Introducción – pág 4
4) Descripción General – pág 4
4.1) Placa Base de la Barrera de Entrada – pág 5
4.2) Placa Base de la Barrera de Salida – pág 9
4.3) Interfaz Gráfica – pág 12
5) Desafíos – pág 16
6) Problemas Encontrados – pág 21
7) Valoración – pág 21
8) Conclusión – pág 22

1) Objetivos

- Integrar de una manera práctica los conocimientos adquiridos a lo largo del curso a partir de la utilización del lenguaje de programación C, desde los primeros pasos en el mundo de los microcontroladores, su programación, utilización de sus periféricos y su comunicación con el mundo exterior.
- Lograr una metodología de trabajo modulada organizada en capas, pudiendo diferenciar fuertemente los drivers, buffers, primitivas y aplicación de manera de que las capas de más bajo nivel sean independiente de las de alto nivel.
- Obtener los conocimientos necesarios para trabajar con los microcontroladores de 32 bits, en particular, el LPC1769 (dispositivo utilizado para llevar adelante el proyecto) y sus periféricos.
- Producir una interfaz gráfica en la PC con la herramienta Qt y el lenguaje C++ que permita recibir y mostrar diversos estados de lo que ocurre en el microcontrolador y a su vez transmitirle estímulos para que éste los interprete y realice las acciones que correspondan.

- Llevar a cabo un enlace entre el microcontrolador y la PC por medio de la comunicación serie donde el flujo de la información se realice de manera bidireccional.

2) Temas Involucrados en Hardware de Cátedra

Tema	Si	No
GPIO	X	
Teclado matricial	X	
Display 7 Segmentos	X	
Display LCD	X	
Display Matricial		X
ADC		X
DAC		X
Más de una UART	X	
Temporizador/Contador	X	
Interrupciones externas		X
Interrupciones de GPIO		X

3) Introducción

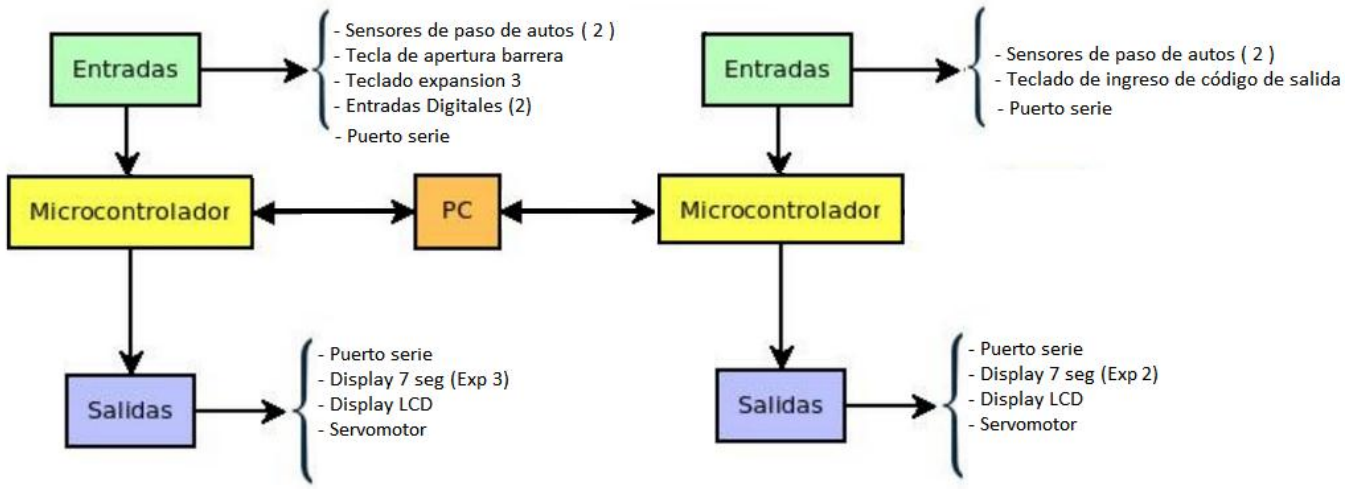
La idea del proyecto consiste en el concepto de una cochera inteligente, un claro ejemplo es la cochera del Shopping Dot. Dicha idea fue impulsada en base a encontrar un proyecto sencillo en cuanto a su dificultad en hardware, en el cual además se pudieran implementar diferentes periféricos con poco hardware adicional y con una conexión directa con una industria que efectivamente posee diferentes soluciones ingenieriles como la que proponemos.

4) Descripción General

El proyecto consta de 2 barreras, una de entrada y una de salida, en el ingreso se obtiene un código el cual se nos exige para poder salir en la barrera de salida. A grandes rasgos se puede dividir en tres partes:

1) **Placa Base de la Barrera de Entrada**
2) **Placa Base de la Barrera de Salida**
3) **Aplicación en la PC**

Diagrama en Bloque:



4.1) Placa Base de la Barrera de Entrada

El conjunto de la placa Base de la Barrera de Entrada consiste en:

Periféricos Externos:

a) Stick lpc 1769
   Stick con microcontrolador 1769 y su jtag acompañante en el cual se carga el programa realizado en LPCXpresso.
b) Placa base Infotronic versión 1.02
   Periférico externo diseñado por la cátedra de Informática II de la UTN FRBA sobre la cual se monta el microcontrolador y el hardware adicional que ya de por si cuenta con hardware adicional.
a) Placa Expansion 3 de 7 segmentos y teclado matricial
   Es una placa con display 7segmentos en la cual se comandan los segmentos a través de unos transistores dirigidos por un integrado bcd comandado por las salidas de gpio del microcontrolador. El teclado matricial sus entradas y salidas también son comandadas o suplidas por entradas y salidas gpio del microcontrolador.
c) Display LCD
   Es un display LCD el cual su integrado se comanda por salidas de gpio como el display 7 segmentos
d) Servomotor Micro Servo SG90
   El Servomotor es utilizado para mover la barrera de entrada, es un servomotor Micro Servo SG 90 de la marca Tower Pro. Es un servomotor con posibilidad de rotación de 0 a 180 grados, tiene un torque de 1,8kgf, una velocidad de rotación de 0,1segundos cada 60º y su tensión de alimentación es de 4,8v.
b) Placa externa con 2 Pulsadores

Basicamente es una placa externa con 2 pulsadores para simular 2 entradas digitales en estado alto o bajo.

e) Código del Microcontrolador llevado a cabo en LPCXpresso
El código del microcontrolador fue realizado en el programa LPCXpresso en el lenguaje de programación C. Dicha herramienta también ofrece la posibilidad de descarga al microcontrolador.

Periféricos lpc 1769:

a) Systick, para llevar una base de tiempo que emplean los timers para conocer el tiempo de estadía.
b) Real Time Clock para llevar un conteo de la hora y fecha actual.
c) Pwm para la señal de control del Servomotor que mueve la barrera de entrada
d) GPIO de salida para comandar periféricos externos como el display LCD o el 7 segmentos y GPIO de entrada para sensar periféricos externos como por ejemplo los pulsadores.
e) UART para una comunicación full-duplex por puerto serie con la pc.

Descripción:

La Placa Base de la Barrera de Entrada posee un Stick lpc 1769 el cual se encarga de controlar todos los periféricos internos y externos al microcontrolador con el programa realizado en LPCXpresso. Dicho Stick se conecta a la Placa Base Infotronic versión 1.02 sobre la cual además se monta la expansión 3, el display LCD, la placa externa con 2 pulsadores y el servomotor Micro Servo SG90.

Funcionamiento:

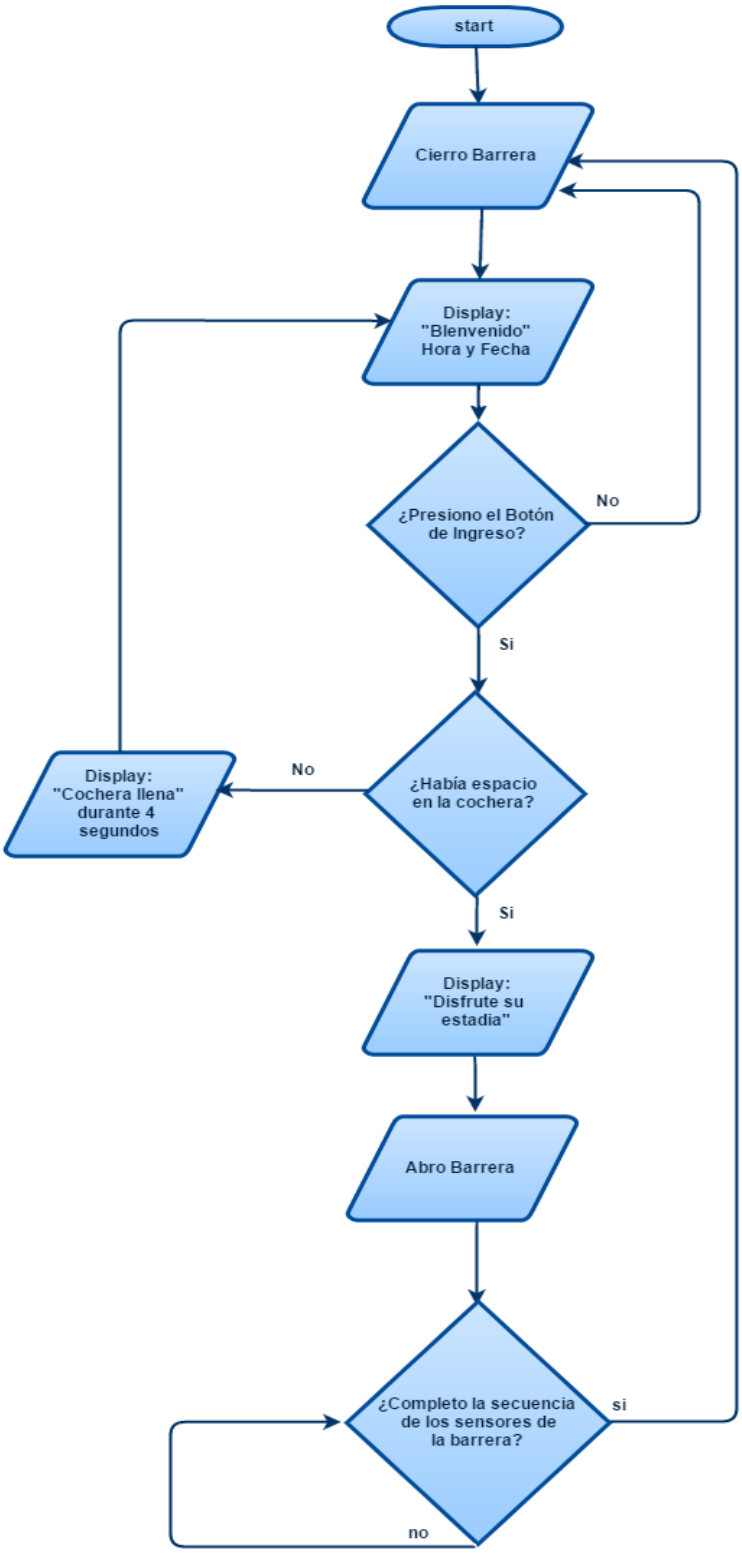
La Placa Base de la Barrera de Entrada posee 5 pulsadores de los cuales se utilizan el switch sw 3 para habilitar el ingreso en la barrera y los switch sw 1 y sw 2 para simular las entradas digitales de los sensores de la barrera de entrada. Previamente a presionar el switch sw 3 la barrera se encuentra mostrando por el display lcd "Bienvenido" seguido de la hora y la fecha. Una vez Presionado el switch sw 3, en el caso de que haya lugar disponible, Se eleva la barrera al moverse el servomotor y se muestra por display lcd "Disfrute su estadía". En el caso de no haber espacio dentro del estacionamiento se muestra por pantalla Cochera llena durante 4 segundos para luego volver al cartel inicial. En el display 7 segmentos se puede visualizar en el primer digito desde la izquierda el estado de la barrera (1 Cerrado, 2 Abierto), en el segundo y tercer digito se visualiza el código correspondiente al ingreso y por último en el digito 5 y 6 se visualiza el estado de la máquina de estados de los sensores de la barrera (Es una máquina de 4 estados). Al completarse la correcta secuencia de los sensores de la barrera da por finalizado el ingreso, se cierra la barrera y vuelve a mostrar el cartel de "Bienvenido" acompañado de la hora y la fecha. A través del teclado Matricial de la expansión 3 y los dos pulsadores de la placa externa se simulan las entradas digitales de los sensores pertenecientes al estacionamiento. Con los switch sw 1 al sw 8 del teclado matricial se ocupa la cochera A1 a

la A8 si estaban previamente desocupadas o se desocupan si estaban previamente ocupadas. En el caso de los pulsadores de la placa externa la lógica es distinta ya que, mientras mantenga presionada la tecla se mantendrá ocupada si estaba desocupada o se mantendrá desocupada si estaba previamente ocupada (Corresponden a las cocheras A1 y A2). A través del periférico de puerto serie y el integrado FT232 se lleva a cabo la conexión con la pc donde se ejecuta la interfaz gráfica. Es una conexión del tipo full-duplex donde ambos miembros envían y reciben datos.

Programa:

La Placa Base de la Barrera de Entrada consiste de 7 aplicaciones principales: Desentramado_Entrada2, Maquina_estados_entrada, Maquina_estacionamiento, Maquina_Estacionamiento_Digital, EstadiaCheck, ActualizaHora Y ActualizaFecha. Desentramado_Entrada2 es una máquina de estados la cual a partir de los datos recibidos por puerto serie en la UART0 toma deciciones y le responde a la interfaz gráfica de ser necesario. Maquina_estados_entrada es la máquina de estados principal de la barrera de entrada, básicamente posee dos grandes estados ESTADO0, barrera cerrada y ESTADO1 barrera abierta, dentro de la cual se ejecuta la máquina de estados de los sensores. Maquina_Estacionamiento la cual corresponde al estacionamiento simulado con el teclado matricial y Maquina_Estacionamiento_Digital la cual corresponde a dos de las cocheras simuladas como verdaderas entradas digitales. EstadiaCheck es la aplicación encargada de llevar a cabo el conteo de los tiempos de estadía de los códigos en la placa base. Por último Actualizahora y ActualizaFecha son las aplicaciones encargadas de llevar a cabo el conteo de la hora y fecha actuales.

Diagrama de Flujo Barrera Entrada



4.2) **Placa Base de la Barrera de Salida**

El conjunto de la placa Base de la Barrera de Salida consiste en:

Periféricos Externos:

f) Stick lpc 1769
Stick con microcontrolador 1769 y su jtag acompañante en el cual se carga el programa realizado en LPCXpresso.

g) Placa base Infotronic versión 1.02
Periférico externo diseñado por la catedra de Informatica II de la UTN FRBA sobre la cual se monta el microcontrolador y el hardware adicional que ya de por si cuenta con hardware adicional.

h) Placa Expansion 2 de 7 segmentos
Es una placa con display 7segmentos en la cual se comandan los segmentos a través de unos transistores dirigos por las salidas de gpio del microcontrolador.

i) Display LCD
Es un display LCD el cual su integrado se comanda por salidas de gpio como el display 7 segmentos

j) Servomotor Micro Servo SG90
El Servomotor es utilizado para mover la barrera de salida, es un servomotor Micro Servo SG 90 de la marca Tower Pro. Es un servomotor con posibilidad de rotación de 0 a 180 grados, tiene un toeque de 1,8kgf, una velocidad de rotación de 0,1segundos cada 60º y su tensión de alimentación es de 4,8v.

k) Código del Microcontrolador llevado a cabo en LPCXpresso
El código del microcontrolador fue realizado en el programa LPCXpresso en el lenguaje de programación C. Dicha herramienta también ofrece la posibilidad de descarga al microcontrolador.

Periféricos lpc 1769

f) Systick, para llevar una base de tiempo que emplean los timers para conocer el tiempo de estadía.
g) Real Time Clock para llevar un conteo de la hora y fecha actual.
h) Pwm para generar la señal de control del Servomotor que mueve la barrera de entrada
i) GPIO de salida para comandar periféricos externos como el display LCD o el 7 segmentos y GPIO de entrada para sensar periféricos externos como por ejemplo los pulsadores.
j) UART para una comunicación full-duplex por puerto serie con la pc.

Descripción:

La Placa Base de la Barrera de Salida posee un Stick lpc 1769 el cual se encarga de controlar todos los periféricos internos y externos al microcontrolador con el programa realizado en LPCXpresso. Dicho Stick se conecta a la Placa Base Infotronic versión 1.02 sobre la cual además se monta la expansión 2, el display LCD y el servomotor Micro Servo SG90.

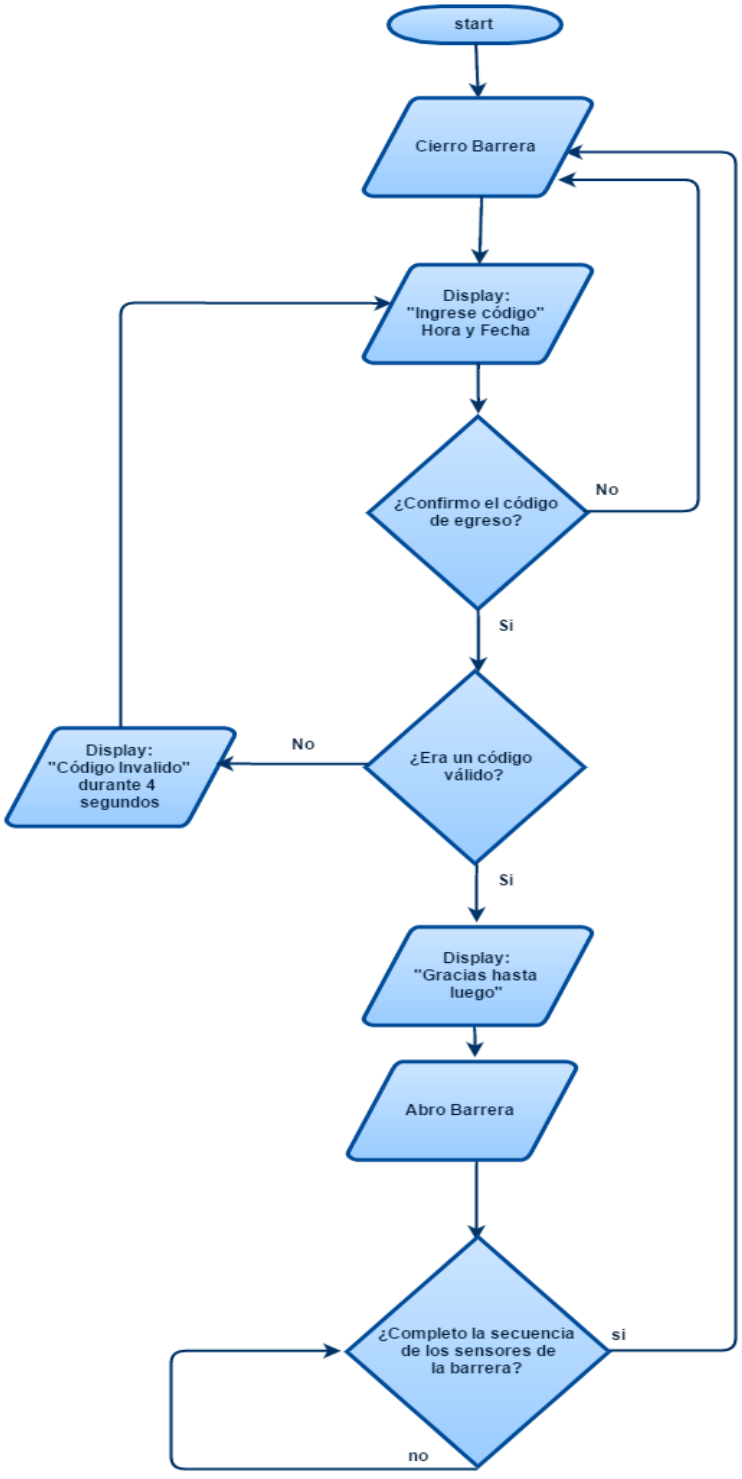
Funcionamiento:

La Placa Base de la Barrera de Salida posee 5 pulsadores de los cuales se utilizan todos. El switch 4 y 3 se utilizan para incrementar y decrementar respectivamente el código mostrado por el display 7 segmentos, el switch 5 para confirmar ese código. Previamente al egreso siempre muestra el cartel "Ingrese Código" acompañado de la hora y fecha. Al ingresar un código valido nos habilita el egreso, caso contrario nos muestra durante 4 segundos código inválido para luego volver al cartel inicial y quedarse a la espera por un nuevo código valido. Al ingresarse un código valido, se abre la barrera y muestra por pantalla "Gracias hasta luego", y comienzan a actuar los switch sw 1 y sw 2 para simular las entradas digitales de los sensores de la barrera de salida. En el display 7 segmentos se puede visualizar en el primer digito desde la izquierda el estado de la barrera (1 Cerrado, 2 Abierto), en el segundo y tercer digito se visualiza el código correspondiente al egreso y por último en el digito 5 y 6 se visualiza el estado de la máquina de estados de los sensores de la barrera (Es una máquina de 4 estados). Al completarse la correcta secuencia de los sensores de la barrera da por finalizado el egreso, se cierra la barrera y vuelve a mostrar el cartel de inicial acompañado de la hora y la fecha. A través del periférico de puerto serie y el integrado FT232 se lleva a cabo la conexión con la pc donde se ejecuta la interfaz gráfica. Es una conexión del tipo full-duplex donde ambos miembros envían y reciben datos.

Programa:

La Placa Base de la Barrera de Salida consiste de 4 aplicaciones principales: Desentramado_Salida_2, Maquina_estados_salida, ActualizaHora Y ActualizaFecha. Desentramado_Salida2 es una máquina de estados la cual a partir de los datos recibidos por puerto serie en la UART0 toma deciciones y le responde a la interfaz gráfica de ser necesario. Por último Actualizahora y ActualizaFecha son las aplicaciones encargadas de llevar a cabo el conteo de la hora y fecha actuales.

Diagrama de Flujo Barrera Salida

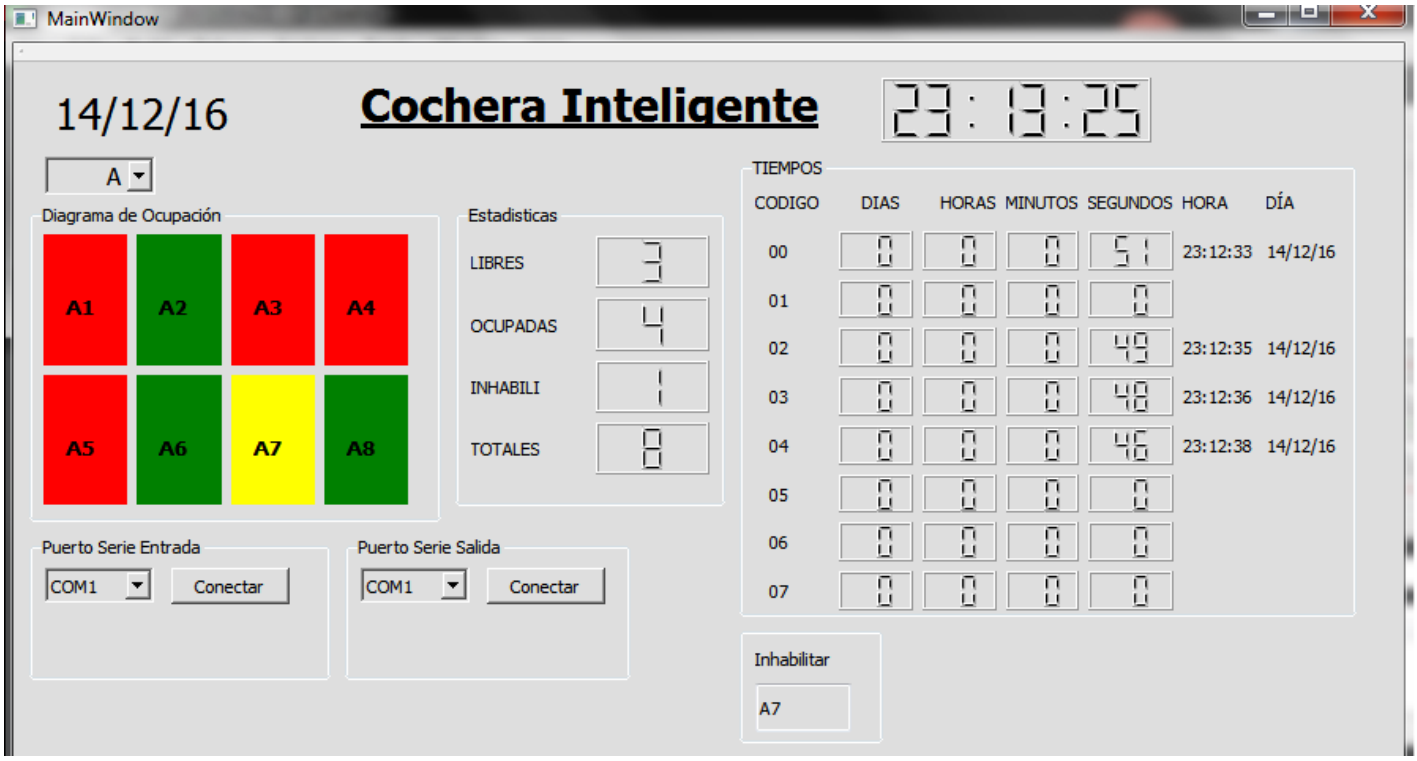


4.3) Interfaz Gráfica en la pc

El conjunto de la aplicación y la pc consiste en:

a) Aplicación llevada a cabo y ejecutada en QtCreator 4.1.0
b) Pc que soporte dicho programa

Imagen:



Descripción:

La aplicación visual de nuestro proyecto permite observar fácilmente el total funcionamiento del estacionamiento automático.

Consta de tres sectores generales:

A la izquierda tenemos el panel de parcelas, en el cual podremos determinar si las mismas están ocupadas, libres o inhabilitadas, todas ellas rotuladas desde la A1 hasta la A8.

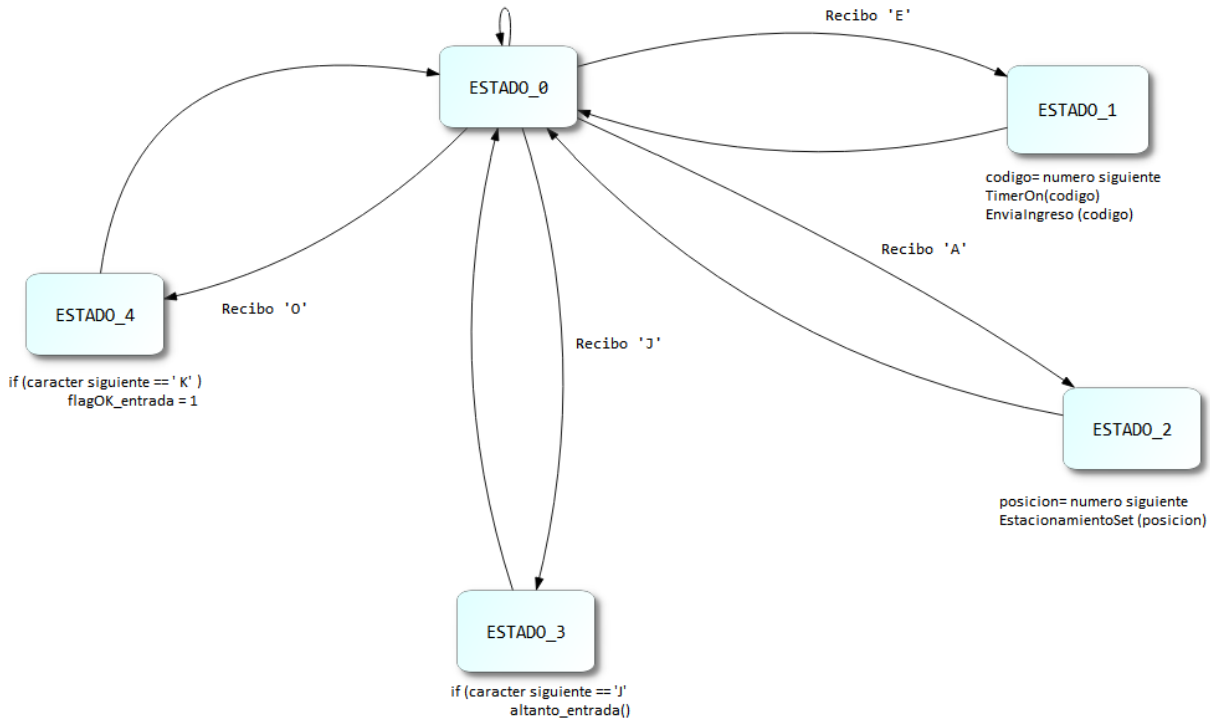
En el centro de la aplicación contamos con un panel de estadísticas, el cual nos facilita información rápida y general acerca del estado actual de nuestra cochera, sobre los códigos que están corriendo en este instante, los lugares libres, las parcelas inhabilitadas y la cantidad total de lugares.

El tercer gran sector de la interfaz gráfica es la zona de timers, ubicada en la esquina superior derecha de la pantalla, en la cual se muestran los tiempos actuales de cada uno de los ocupantes de la cochera. Además de suministrarnos el tiempo exacto de ocupación, dividido en segundos, minutos, horas y días, también nos indica la hora precisa en la cual ingresó cada uno de los autos.

En cuanto a la relación con los dos microcontroladores (uno en la barrera de salida y otro en la barrera de entrada) la interfaz gráfica actúa como maestra, siendo la que comienza la comunicación serie cuando las placas estén disponibles, enviándoles toda la información que necesitan para su correcto funcionamiento.

Máquinas de estado de Desentramado en la aplicación:

Desentramado Entrada:



Esta máquina consta de 5 estados, los cuales llamamos estado 0, 1, 2, 3 y 4 para mayor comodidad.

Como lo indica su nombre, la tarea que desarrolla es la de recibir, procesar y decidir las acciones a tomar dependiendo de qué información reciba desde el puerto serie de la barrera de entrada.

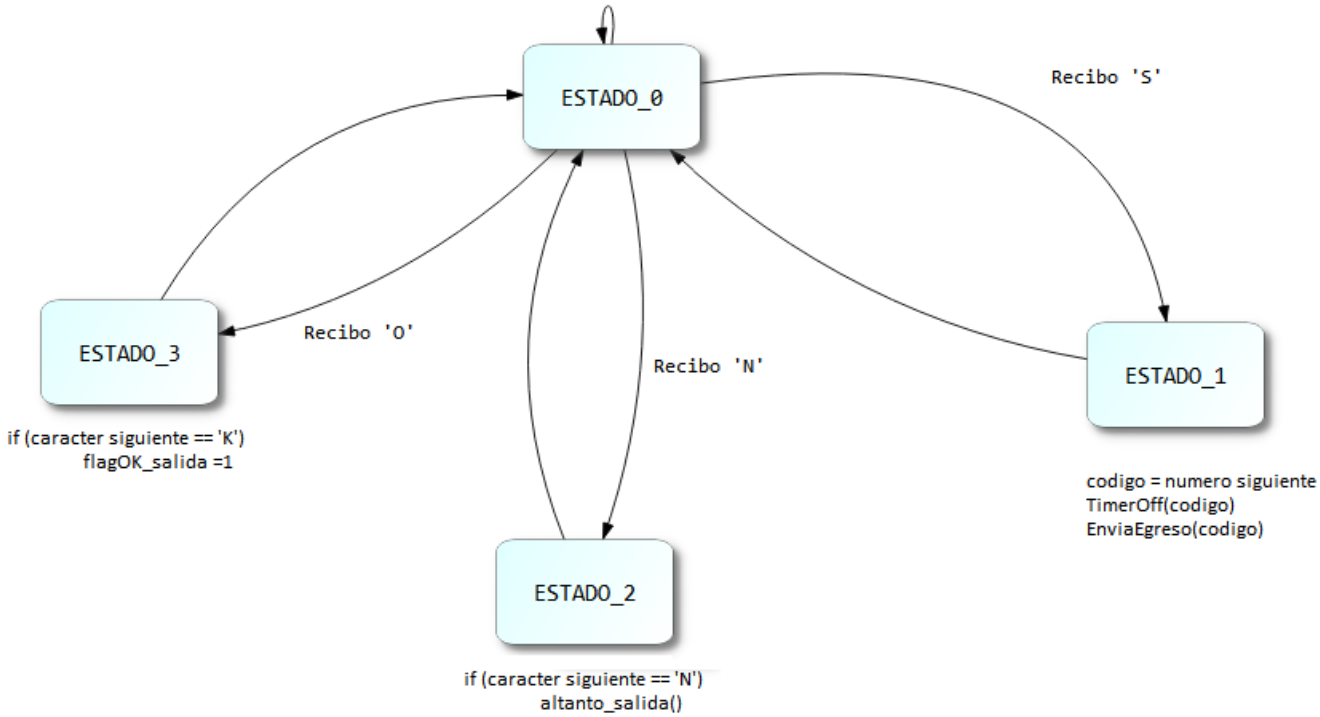
Si recibo un carácter 'E' de la placa de entrada me está notificando que acaba de ingresar un auto, pasaremos al Estado 1, en el cual evaluaremos el número que sigue al carácter E, el cual indica el número de código que se le otorgó al nuevo vehículo (Dichos códigos son del 0 al 7). Llamamos a la función TimerOn, la cual inicia timers y realiza todas las acciones de ingreso pertinentes y mediante la función EnviaIngreso() avisaremos por puerto serie a la placa de salida el nuevo código para que lo tome como válido a la hora de que el vehículo decida retirarse. Por ultimo regresamos al Estado 0 a la espera de un nuevo caracter

Si recibo un carácter 'A' significa que hubo un cambio en el estado de alguno de los sensores de presencia de las cocheras, pasaremos al Estado 2 y le enviaremos el número que sigue al carácter ‘A’ a la función EstacionamientoSet() la cual realizara las acciones que correspondan, ocupando y desocupando las parcelas de la interfaz. Por ultimo regresamos al Estado 0.

Si recibo un carácter 'J' pasaremos al Estado 3 y constaremos de que el carácter siguiente también sea una 'J', de ser así, significa que la placa de entrada necesita una actualización de datos, probablemente debido a que sufrió un corte de energía. Llamamos a la función altanto_entrada() que se encargará de enviar toda la información necesaria respecto de los códigos activos, inhabilitados, desocupados, la fecha y la hora. Por ultimo regresamos al Estado 0.

Si recibo un carácter 'O' pasaremos al Estado 4 y si el caracter siguiente es una 'K', significa que la placa de entrada ha recibido sin problemas la actualización y ya no es necesario que sigamos enviándosela. Dejamos esto constatado en el flagOK_entrada y regresamos al Estado 0.

Desentramado Salida:



Esta máquina consta de 4 estados, los cuales llamamos estado 0, 1, 2 y 3 para mayor comodidad.

Como lo indica su nombre, la tarea que desarrolla es la de recibir, procesar y decidir las acciones dependiendo que información reciba desde el puerto serie de la barrera de salida.

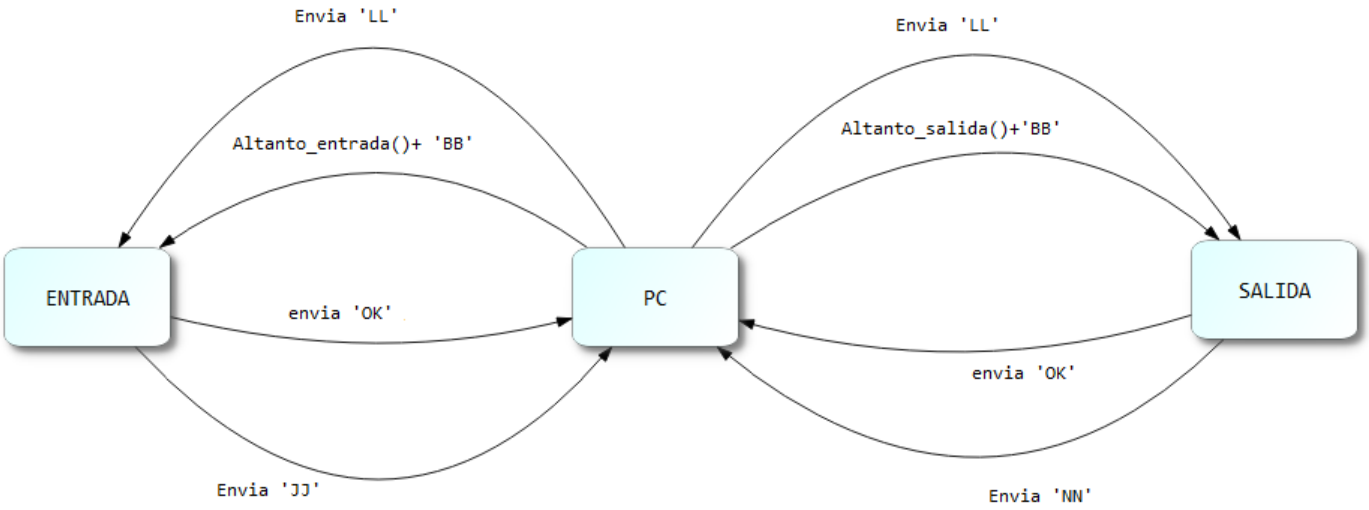
Si recibo un carácter 'S' significa que acaba de irse un auto, por ende hay que liberar el código para reutilizarlo. Consecuentemente pasaremos al Estado 1, en el cual se finaliza el timer correspondiente en la interfaz y se avisa por puerto serie a la placa de entrada que el código esta nuevamente disponible para ser usado. Por último se vuelve al Estado 0.

Si recibo un carácter 'N' pasaremos al Estado 2, en el cual evaluaremos si el siguiente carácter también es una letra 'N' en cuyo caso, significa que la placa necesita actualizar la información que posee, posiblemente por algún corte de energía que tuvo o primer encendido. Acto seguido se llama a la función Altanto_salida, la cual se encarga de enviarle por puerto serie

toda la información que necesita para su óptimo funcionamiento. Por último se vuelve al Estado 0.

Si recibo un carácter 'O' pasaremos al Estado 3 en el cual evaluaremos si el siguiente carácter es una 'K', en cuyo caso significa que la placa de salida ha podido actualizar sus datos sin inconvenientes y que no es necesario volver a enviárselos y lo dejamos constatado en el flagOK_salida. Por último se vuelve al Estado 0 a la espera de un nuevo carácter.

Secuencia de Actualización



5) Desafíos

Nuestros desafíos consisten en el aprendizaje y correcta aplicación de los distintos periféricos empleados en el proyecto, haciendo hincapié en los periféricos no vistos en clase, como el RTC, el PWM, los periféricos externos como el teclado matricial, el display 7 segmentos comandado por un integrado bcd, el display LCD, 2 entradas digitales externas, así como también todos los desafíos a nivel aplicación como hacer una aplicación autónoma, con buckup a nivel archivos frente a perdida de conexión o corte de luz sin importar la secuencia de conexión de las placas con la pc, ni cuál fue el que perdió la conexión o la alimentación (Placa entrada, Placa Salida o PC).

Algunas funciones a destacar:

Activos

```
void MainWindow:: Activos (void)
{
    static int estadoAnt[8]={0};
    int i,j,flagCambio=0,CocheraNumero;


    for(i=0;i<8;i++)
    {
        if(estadoAnt[i] != vectorFlags[i])//si acaba de entrar o salir alguien
        {
            flagCambio=1;
            estadoAnt[i]= vectorFlags[i];
        }
    }
    if(flagCambio==1)
    {
        fs.open("Activos.txt");
        for(i=0;i<8;i++)
        {
            if (vectorFlags[i]==1)//alguien acaba de entrar
            {

                switch(i)
                {
                    case 0:
                        fs << (char)((int)i + 48) <<" "<< ui->label_00_hora->text().toStdString();
                        fs   << ":" << ui->label_00_dia->text().toStdString() << endl;
                        break;
                    case 1:
                        fs << (char)((int)i + 48) <<" "<< ui->label_01_hora->text().toStdString();
                        fs   << ":" << ui->label_01_dia->text().toStdString() << endl;
                        break;
                    case 2:
                        fs << (char)((int)i + 48) <<" "<< ui->label_02_hora->text().toStdString();
                        fs   << ":" << ui->label_02_dia->text().toStdString() << endl;
                        break;
                    case 3:
                        fs << (char)((int)i + 48) <<" "<< ui->label_03_hora->text().toStdString();
                        fs   << ":" << ui->label_03_dia->text().toStdString() << endl;
                        break;
                    case 4:
                        fs << (char)((int)i + 48) <<" "<< ui->label_04_hora->text().toStdString();
                        fs   << ":" << ui->label_04_dia->text().toStdString() << endl;
                        break;
                    case 5:
                        fs << (char)((int)i + 48) <<" "<< ui->label_05_hora->text().toStdString();
                        fs   << ":" << ui->label_05_dia->text().toStdString() << endl;
                        break;
                    case 6:
                        fs << (char)((int)i + 48) <<" "<< ui->label_06_hora->text().toStdString();
                        fs   << ":" << ui->label_06_dia->text().toStdString() << endl;
                        break;
                    case 7:
                        fs << (char)((int)i + 48) <<" "<< ui->label_07_hora->text().toStdString();
                        fs   << ":" << ui->label_07_dia->text().toStdString() << endl;
                        break;
                }

            }
            if(vectorFlags[i]==0)//!<Desocupado
            {
                fs <<(char)((int)i + 48)<<" "<<"libre" << "------------" << endl;
            }
            if(vectorFlags[i]==2)//!<Esta inhabilitado, por ende desocupado
            {
                for(j=0;j<8;j++)
                {
                    if(aux[j]==i)
                    {
                        CocheraNumero=j+1;

                    }
                }
                fs <<(char)((int)i + 48)<<" "<<"inhabilitado" << "-A" << CocheraNumero << "--" << endl;

            }
        }
        flagCambio=0;
    }
    fs.close();
}
```

Código de PWM
Inicialización

```
void InicializarPWM ( void )
{
    PCONP = PCONP | (1<<6);     // Energizo el PWM1

    // Selecciono el PCLK del PWM1: PCLK_PWM1 = CCLK
    PCLKSEL0 = PCLKSEL0 & ( ~(3<<12) );
    PCLKSEL0 = PCLKSEL0 | (1<<12);

    // Le doy la Funcionalidad de los Pines como Salidas/Canales de PWM.
    SetPINSEL (PWM2, PINSEL_FUNC1);     // Giro en Sentido Horario. *****  MARCHA
    SetPINSEL (PWM3, PINSEL_FUNC1);     // Giro en Sentido Antihorario.  ******* REVERSA
    SetPINSEL (PWM4, PINSEL_FUNC1);//led rojo

    // Selecciono a los Canales de PWM como Salidas.
    SetDIR (PWM2, SALIDA);
    SetDIR (PWM3, SALIDA);

    PWM1PR = PWM1_PRESCALER;        // Base de Tiempo: microsegundos.

    // Selecciono la Frencuencia del Motor:
    PWM1MR0 = FRECUENCIA_GIRO;              // Frecuencia de Giro del Motor: 250Hz

    // Selecciono los Valores del Duty Cycle (Tiempo de Trabajo):
    PWM1MR2 = PWM1_MATCH2;
    PWM1MR3 = PWM1_MATCH3;

    // Para que Interrumpa y Resetee el Timer cuando hay un Evento por MATCH0 e Interrumpa por MATCH2 y MATCH3
    PWM1MCR = PWM1MCR | 0x00000243;

    // Registro para Actualización de los Valores de MATCHx (Actualización de MATCH0, MATCH2, MATCH3).
    PWM1LER = PWM1LER | 0x0000000C;

    // Habilito Canales de Salida
    PWM1PCR = PWM1PCR & ( ~(1<<2) );    // Salida PWM2: Simple Edge
    PWM1PCR = PWM1PCR | (1<<10);    // The PWM2 Output Enabled.
    PWM1PCR = PWM1PCR & ( ~(1<<3) );    // Salida PWM3: Simple Edge
    PWM1PCR = PWM1PCR | (1<<11);    // The PWM3 Output Enabled.

    PWM1TCR = PWM1TCR & 0x00000004;     // Limpio los Bits de Timer/Counter, Reset, PWM1 Enable.
    PWM1TCR = PWM1TCR | 0x02;       // Reseteo el Timer.
    PWM1TCR = PWM1TCR & (~(0x02));  // Limpio Bit de Reset.

    // Habilito el Counter/Timer y Habilito el PWM1. ===> Counter Enable Y PWM Enable.
    PWM1TCR = PWM1TCR | 0x00000009;

    ISER0 = ISER0 | (1<<9);     // Habilito la Interrupción del PWM en el NVIC.
}
```

PopServo y PushServo

```
void PopServo ( void )
{
    uint32_t aux [ CANT_CANALES_PWM ];

    aux [ 0 ] = Buff_PWM [ 0 ];

    aux [ 1 ] = Buff_PWM [ 1 ];

    PWM1MR2 = aux [ 0 ];

    PWM1MR3 = aux [ 1 ];

    PWM1LER = PWM1LER | 0x0000000C;
}
```

```
void PushServo (uint8_t angulo)
{
    switch ( angulo )
    {
        case CEROGRADOS:
            Buff_PWM [ 0 ] = ( (FRECUENCIA_GIRO * 4) / 100 );//0º//( (FRECUENCIA_GIRO * 25) / 100 );
            Buff_PWM [ 1 ] = VALOR_NULL;
            break;

        case NOVENTAGRADOS:
            Buff_PWM [ 0 ] = ( (FRECUENCIA_GIRO * 16) / 200 );//( (FRECUENCIA_GIRO * 60) / 100 );
            Buff_PWM [ 1 ] = VALOR_NULL;
            break;

        case CIENTOOCHENTA:
            Buff_PWM [ 0 ] = ( (FRECUENCIA_GIRO * 20) / 100 );//180º
            Buff_PWM [ 1 ] = VALOR_NULL;
            break;


        default:
            Buff_PWM [ 0 ] = ( (FRECUENCIA_GIRO * 4) / 100 );//0º
            Buff_PWM [ 1 ] = VALOR_NULL;
            break;
    }
}
```

Código RTC

```
void Inic_RTC(void)
{
    RTCILR = 0x00;              //!< Limpio Flags
    RTCCIIR = 0x00;             //!< Deshabilito interrupciones
    RTCAMR = 0xFF;              //!< Enmascaro comparador de alarmas
    RTCCALIBRATION = 0x00;      //!< Inicializa registro de calibración
    RTCCCR = 0x01;              //!< Arranca contador
}

    \fn void RTC_HabilitaInterrupcion(uint32_t tipo_int)

void RTC_HabilitaInterrupcion(uint32_t tipo_int)
{
    RTCCIIR |= 1<< tipo_int;    //se habilita el tipo de interrupcion pedido (MODIFICO CADA CUANTO QUIERO QUE ME INTERRUMPA
    //                              1 SEGUNDOS 10 MINUTOS 100 HORAS 1000 DIAS
    RTCILR = 0x03;              //limpio flags de interrupcion
    ISER0 |= 1<<17;             //habilito interrupcion desde el NVIC
}



    \fn void RTC_IRQHandler(void)


void RTC_IRQHandler(void)
{
    RTCILR |= 0xFF;     //!<limpio todos los flags de interrupcion
    f_tick_rtc = 1;
}
```

6) Problemas Encontrados

Consumo del pwm:

Uno de los principales problemas que encontramos durante el desarrollo del proyecto fue que el consumo del servomotor es elevado entonces dependiendo del Output de la fuente de alimentación, conectarlo directamente con la placa genera problemas e impide el correcto funcionamiento del programa.

Para solucionarlo era claro que necesitábamos recurrir a una fuente de corriente externa que alimentara el motor.
Para evitar la utilización de una fuente regulable, optamos por alimentar la placa de dos formas al mismo tiempo, además de utilizar la fuente de 12 v conectamos el cable USB de debugueo al stick, el cual nos provee de la energía necesaria para el correcto funcionamiento del proyecto sin mayores modificaciones.

Utilización de LCD:

Otro inconveniente que tuvimos fue al utilizar el display LCD tanto en la placa de entrada como en la de salida, ya que sus características bloqueantes, anulaban muchos pulsos de teclas y procesos internos, tornando muy poco confiable la aplicación. Frente a esto, nos vimos obligados a cambiar una estrategia de programación mediante flags para lograr el correcto funcionamiento del proyecto, que consiste en tratar de usar la función para mostrar por display lo menos posible.

7) Valoración

Al realizar este proyecto logramos integrar y reforzar muchos de los conocimientos adquiridos en la materia y, a su vez, integrar otros de otras materias relacionadas más con el área de hardware de la electrónica. Dentro de lo referido a la materia se logró reforzar y mejorar día a día todo lo aprendido en cuanto al desarrollo y a las estrategias y técnicas de programación como por ejemplo la implementación en capas, el diseño e implementación de máquinas de estados para resolver procesos y cómo optimizar mediante software distintas tareas con distintas soluciones.

A nivel aprendizaje, la mayoría de los nuevos conocimientos que se adquirieron son fácilmente aplicables a otras materias o problemas de la vida laboral de un ingeniero o técnico, inclusive en ámbitos estudiantiles y de otras carreras o áreas laborales.

8) Conclusión

Se pudieron cumplir la mayoría de los objetivos planteados en el inicio de este documento, los que no se llegaron a cumplir están prácticamente a un paso y no se llevaron a cabo por falta de tiempo y no hacen a la esencia del proyecto.
Se pudo hacer una utilización práctica tanto de casi todos los tópicos vistos a lo largo del curso, además de la inclusión de temas no vistos incorporados como desafíos.
Se logró el objetivo de producir una interfaz gráfica en PC que muestre el estado y funcionamiento del estacionamiento inteligente en tiempo real, sea autónoma y esté protegida contra problemas de energía, resets o problemas de comunicación, los cuales son problemas reales del uso cotidiano de aplicaciones.
Se logró producir una comunicación serie robusta, desde la elección de la forma de envío de datos hasta su recepción, procesamiento y respuesta, empleando comunicación full dúplex entre aplicación y placas, siendo la interfaz el sistema principal sobre el cual se apoyan las dos placas.