### Alumno: Pablo Daniel Folino

### Plataforma embebida: EDU-CIAA-NXP

### Aplicación:

El sistema (plataforma robótica) posee como entrada dos teclas que se pueden pulsar en forma alternada y como actuadores dos motores de corriente continua conectados a dos PWM de la placa EDU-CIAA. Una tecla se utilizará para realizar un movimiento hacia adelante y la otra hacia atrás. Cada vez que se pulsa un de ellas se mide la cantidad de tiempo en que se mantiene presionada, y al soltarla realizará el movimiento correspondiente. Se informa mediante un par de leds el estado en que se encuentra el robot.

La velocidad de desplazamiento (valor de PWM) será un valor fijo.

### Periféricos :

PWM, GPIO

### Diagrama de estado de MEF con una breve descripción de cada estado.

SLEEP

* fsmButtonInit(estadoPulsador\_t \*) inicializa la máquina de estados de los pulsadores.

P

Reset

SLEEP

* fsmButtonInit(estadoPulsador\_t \*) inicializa la máquina de estados de los pulsadores.

P

SLEEP

* fsmButtonInit(estadoPulsador\_t \*) inicializa la máquina de estados de los pulsadores.

P

READY\_DOWN

Timer0ador\_t \*) inicializa la máquina de estados de los pulsadores.

P

READY\_DOWN

Timer1lsador\_t \*) inicializa la máquina de estados de los pulsadores.

P

### 

READY\_DOWN

Timer0lsador\_t \*) inicializa la máquina de estados de los pulsadores.

P

READY\_DOWN

Timer1lsador\_t \*) inicializa la máquina de estados de los pulsadores.

P

### Descripción de cada estado

### Inicio: Se configura el estado inicial del sistema.

### ESPERA: en este estado el sistema espera que se presione una tecla, y cuando sucede mide el tiempo que está presionada.

### ADELANTE: mueve la plataforma robótica hacia adelante por una cierta cantidad de tiempo.

### ATRAS: mueve la plataforma robótica hacia atrás por una cierta cantidad de tiempo.

### Definir los módulos de software (archivos) que va a implementar para cada periférico.

Robot\_RL.c  y Robot\_RL.h 🡪 módulo principal (main) que lanza la aplicación y controla los tiempos de ejecución del módulo de la FSM principal. Se implementa un sistema de tiempo real cooperativo con una base de tiempos de 1mseg.

fsm\_robot.c y fsm\_robot .h 🡪 máquina de estados principal de control de robot.

fsm\_button.c y  fsm\_button.h 🡪 máquina de estados para monitorear los dos pulsadores de entrada (anti-rebotes).

fsm\_timer.c y fsm\_timer.h 🡪 lleva la cuenta de cuánto tiempo son presionadas las teclas. Los hay dos contadores uno para la tecla que determina el movimiento hacia adelante y otro para la tecla hacia atrás. Cada contador posee dos campos, en uno se lleva la cuenta, y en el otro se registra el estado del contador, los cuales pueden ser: SLEEP, READY\_UP, y READY\_DOWN.

led.c  y led.h 🡪 para manejar los leds que informan el estado actual. EL estado se informará con dos led´s indicando si se mueve la plataforma hacia adelante, hacia atrás, o se encuentra en estado de reposo.

motor.c y motor.h 🡪    para actuar sobre los dos motores de corriente continua.

Definir los prototipos de las principales funciones públicas y privadas de cada módulo definido.

fsm\_robot .h

* fsmRobotInit(void \* TaskParam) inicializa las condicione iniciales (estado en ESPERA) del robot. Llama a las funciones de inicialización de los motores.
* fsmRobotUpdate(void \* TaskParam) actualiza la máquina de estados del robot. Los estados posibles son: ESPERA, ADELANTE, ATRÁS, NN(para futuras ampliaciones).

fsm\_timer.h

* fsmTimerInit (void \* TaskParam) inicializa los contadores de tiempo.
* fsmTimerUpdeta(void \* TaskParam) máquina de estados de los contadores.

fsm\_timer.c

* TimerUpdate (estadoPulsador\_t \*, uint8\_t contador ) evalúa los estados de los pulsadores accediendo a los estdos de los mismos mediante un vector.

fsm\_button.h

* fsmButtonInit(estadoPulsador\_t \*) inicializa la máquina de estados de los pulsadores.
* fsmButtonUpdate(void \* TaskParam) actualiza los estados de los pulsadores en un vector. Los estados posibles son: B\_ON, B\_OFF, B\_RISING, y B\_FILLING.

fsm\_button.C

* ButtonUpdate(estadoPulsador\_t \* vectorTeclas, uint8\_t numeroTecla, bool\_t estadoTecla) evalúa el estado de cada tecla.

motor.h

* bool motorSpeed(int \*valor, bool Motor ) configura la velocidad del motor. Devuelve “0” si todo ok, y si “1” la velocidad no está comprendida entre un mínimo y un máximo. Si Motor vale “1” estoy modificando la velocidad del motor de la derecha, y si vale “0” es el de la izquierda.
* motorSpeedMax(int \*valor) configura la máxima velocidad de los motores(es un limitador).
* motorSpeedMin(int valor) configura la mínima velocidad de los motores(es un limitador).
* motorEnable(bool Enable/Disable,, bool Motor) enciende o apaga uno de los motores.

motor.c

* motorSpin(bool valor, bool Motor ) configura el sentido de dirección de un motor . Se usan cuatro puertos gpio, dos para cada motor.