Universidad de Castilla-La Mancha
Escuela Superior de Informática (CR)

Proyecto 2

Micro-Python con STM32-F411RE

LED-Dimmer

Sergio Jiménez del Coso
Profesor: Julián Caba Jiménez

Diseño de Sistemas Basados en Microprocesadores
7 de junio de 2020

Índice

Índice de figuras

1. Introducción

Con la realización de este trabajo se pretende poner en práctica todos los conocimientos tanto teóricos como prácticos que he adquirido a lo largo de la asignatura de Diseño de Sistemas Basados en Microcontralodores.

Para la resolución de este proyecto, estableceré un diseño previo para determinar una solución de manera sencilla y eficiente. También se pone en práctica la capacidad de síntesis, de cara a la obtención de una solución en conjunto.

Además, en este proyecto incluyo una variante a la hora del desarrollo del proyecto: la implementación a través de un nuevo lenguaje de programación, MicroPython. Este lenguaje permite que la implementación sea mucho más sencilla y sobre todo mucho más cómoda a la hora de escoger las librerías necesarias para cada uno de los pines asociados. Finalmente, utilizaré el microcontrolador STM32-F411RE, el cual he estado familiarizado a lo largo de esta asignatura.

2. Definición del proyecto

Este proyecto consiste en el cambio de la luminosidad de un LED a través de un sensor de ángulo rotativo (*rotary angle sensor*), es decir, podemos hablar de un **LED dimmer**. Una vez que giremos el sensor de ángulo rotativo, la intensidad lumínica del LED varía, dependiendo del ángulo del sensor. A continuación definiré cada uno de los contenidos que son importantes de cara a la resolución del diseño de este proyecto.

3. Diseño del proyecto

3.1. ADC, PWM, Timer

Para realizar el diseño de este proyecto, primero observé qué necesitaría para llevar a cabo una solución. Para empezar, el valor del sensor de ángulo rotativo era necesario una señal capaz de transformar un valor analógico a digital, es decir, el **ADC** (Analog to Digital Converter). En este caso, la resolución del *ADC* es de 12 bits, ya que está por defecto en MicroPython. Para resolver el problema del LED, no se trata de encenderlo y apagarlo, sino asociarle una señal que permita cambiar la intensidad lumínica y modificarla con el valor recogido del *ADC*. Para ello utilizaré una señal **PWM** (Pulse Width Modulation), ya que se trata de la cantidad de tiempo del LED en cada ciclo. Ahora tenemos que definir cada uno de los pines que vamos a utilizar:

Input/Output	Pin	Señal
Rotary Angle Sensor	PA1	ADC
LED	PA5	PWM

Cuadro 1: Pines asociados

De forma gráfica lo podemos ver en el Fritzing:

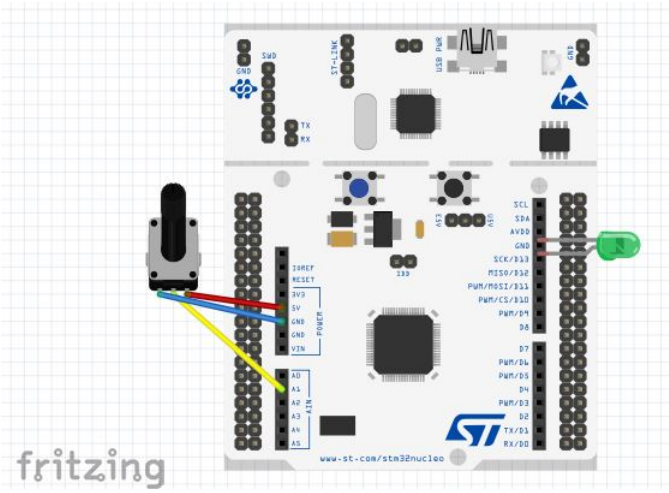

Figura 1: Diseño en Fritzing

No obstante, el problema que tiene MicroPython con esta placa es que el pin que tenemos asociado al *rotary angle sensor*, el cual está conectado a un *ADC*, no presenta ningún tipo de interrupción de cara a la lectura de nuestro sensor rotativo. En resumen, ningún pin asociado a una señal *ADC* permite leer el valor a través de una interrupción definida. Para solucionar dicho problema, he inicializado un Timer capaz de saltar una interrupción con el objetivo de leer el valor de nuestro sensor, y así modificar la señal de nuestro LED para que la intensidad varíe. El Timer que he tenido que utilizar es el Timer 5, ya que el Timer 6, el cuál es el ideal para las señales *ADC*, no está disponible en nuestra placa. La interrupción saltaría después de cada variación del valor del *Timer*.

Al no disponer de ninguna placa *Protoboard*, he tenido que conectar el LED directamente al pin correspondiente. No obstante, para no causar ningún daño en él ni fundirlo, la intensidad aplicada a la señal no es alta. Será un valor que sea capaz de visualizar las variaciones de la intensidad.

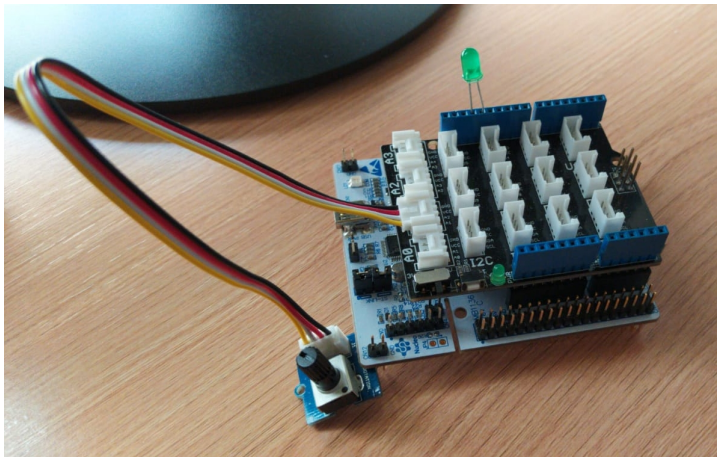

Figura 2: LED-dimmer

3.2. Final State Machine

Para mejorar la funcionalidad del proyecto, he tenido que diseñar una máquina de estados. Esta máquina de estados será simple ya que disponemos de un actuador que es el LED y un dispositivo de entrada que es el sensor rotativo.

Esta máquina de estados, aparte de mejorar el rendimiento y la funcionalidad de nuestro proyecto, nos ayuda a entender cómo funciona nuestro proyecto. Cuando el timer haya sido inicializado, para la captura de las interrupciones, la máquina de estados alternará entre los dos últimos estados como si fuera un bucle infinito. En el caso de que el usuario pulse el botón de *RESET* que está asociado a la placa, el sistema finalizará. En la Figura 3, podemos observar cada uno de los estados, los cuales definiré a continuación:

- **Init Timer**: En este estado iniciamos el timer que permite capturar cada una de las interrupciones para poder leer el valor del sensor rotativo.
- **Change brightness (Cambio de intensidad lumínica)**: Una vez que haya capturado la interrupción para obtener el valor de nuestro sensor rotativo, se calcula la intensidad con la que va a lucir nuestro

LED. Se multiplicará por un número entero para que se pueda percibir con claridad dicha intensidad.

- **LED dimmed (LED oscurecido)**: En este estado modificamos el valor de la señal que está asociada al LED.

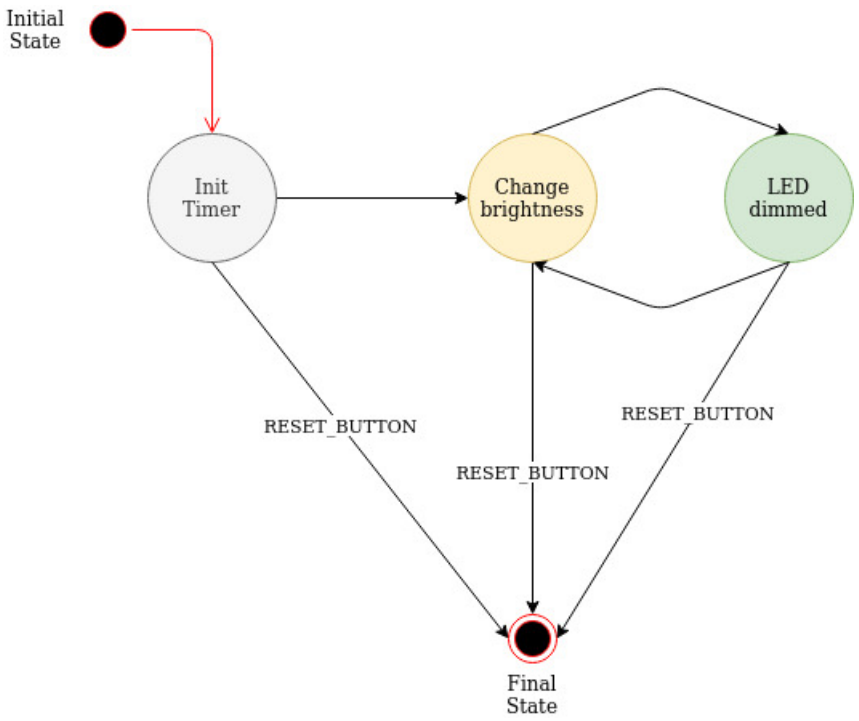

Figura 3: Máquina de Estados

4. Conclusión

En conclusión, a la hora del desarrollo de este proyecto pongo en práctica todos los conocimientos que he adquirido a lo largo de esta asignatura y sobre todo realizarlo con un lenguage de programación que es muy sencillo de utilizar como es el caso de MicroPython.

Además, a la hora de la implementación es mucho más sencilla, ya que se trata de un lenguaje de programación que simplifica cada una de las funcionalidades en este proyecto. No obstante, he tenido que buscar otra funcionalidad ya que como he dicho anteriormente, se trata de un lenguaje mucho menos extenso y con menos funcionalidades de las que ofrece C. No obstante, al final se puede adaptar una solución muy similar que es lo que se ha realizado en este proyecto.

Finalmente, ha sido un proyecto en el que he aprendido que no solo existe un lenguaje que pueda satisfacer dichas funcionalidades y más si se trata de un proyecto cuyas funcionalidades se pueden realizar de forma sencilla, como ocurre en este caso.

5. Referencias

1. *MicroPython for the Internet of Things A Beginner's Guide to Programming with Python on Microcontrollers*, Author: *Charles Bell*

2. Micro-Python Documentation