Laboratorio n°3

Se utiliza la placa galileo de primera generación. Es un procesador con un monohilo, es un system on chip. Tiene pines arriba para que se pueda empalmar con arduino.

Actividad 1: parte de introducción a esta placa.

Vamos a utilizar una versión vieja de IoT. El entorno que vamos a utilizar esta en C:\programs\iot…\...bar

1. Conectamos la plaquita, abrimos putty, poenmos la dirección, y accedemos con root. Nos va a tirar una consolita

Nos va a abrir una consolita, que es como un sistema operativo, podemos hacer ps, ls sobre este sistema.

3. Creamos un proyecto en Iot, seleccionado crear proyecto en IoT, con la opción Blink led C++

Cuando creamos con new, Intel Iot C/c++ proyect, elegimos on board Led blink C++ y después tiraba una ventana de conección, había que escribir root.

Estamos viendo que si usamos dos máquinas para una sola placa, si creamos dos proectos, al ser monohilo, atiende solo a uno de los proyectos a la vez. Tiene un sistema de archivos la placa podemos eliminar archivos y agregar, etc.

Lo que hace este programa es apagar y prender un LED cada un segundo.

4. Desde la consolita SSH que abre putty, vamos a la carpeta donde tenemos el rpoyecto y ejecutamos ./nombre del proyecto, y vamos a ver como se apaga y se prende el LED en la plaquita.

Podemos ver aca que es un proceso corriendo este programita, entonces si matamos el proceso mientras el LED esta prendido o apagado, este LED queda como quedo en el último estado, osea o prendido o apagado

5. Utilizando el IDE de Arduino, cargamos el ejemplo de arduino en el propio arduino, vemos que este lo que hace es mediante el cable, enviar el mensaje de hello, y supuestamente el receptor deberá tomar dicho mensaje (en este caso la galileo, utilizando la librería mraa) e imprimirá dicho mensaje por alguna salida, en este caso la galileo por pantalla de la pc en el IDE de IoT.

6. Descargamos el master\_receive de la página, y luego lo cargamos en el IoT. Lo vamos a cargar igual que el proyecto de blink anterior.

7. Conectamos el arduino a la placa galileo, grabamos en el arduino el código de slave\_send y ejecutamos el código de master\_receive en el galileo con IoT, y vamos a ver por el serial de IoT que recibe continuamente “Hello” del arduino.

En cuanto a la librería mraa:

En arduino: En el address, le vas a decir el esclavo con el cuál

Wire.begin(8); acá le dice que su dirección es 8, y es esclavo si tiene dirección. Si no tiene dirección es el master. Join i2c bus with address #8

Wire.onRequest(requestEvent); register event, acá le asigno la función de callback, como hacíamos en el laboratorio anterior con el igual a mi\_callback(); que está declarada en la librería de Wire, la va a andar llamando desde ahí, cuando en este caso galileo de IoT, hago el read y ahí le pide que reciba algo, le envía las s

Luego arduino, wire tiene declarada la función de callback y galileo se encarga de llamarla, adentro de wire hace toda la lógica para mapearla.

En galileo, en cambio, cuando lo creo con IoT hago:

i2c = new mraa:I2C(0); creo la I2c y con el 0 indica que es master

i2c->address(8); le asigno la dirección del esclavo, en este caso 8, que es la que le defini del arduino, la puedo cambiar mas adelante en el código si es que quiero utilizar dos slaves.

i2c->read(rx\_tx\_buf,6); Cuando galileo hace read, manda por I2C la solicitud, le pasa como parámetro el buffer donde voy a guardar y le escribe el tamaño a recibir. Y esta solicitud se envía por el bus de I2c hacía el slave, en este caso el arduino, le va a llegar por la callback de RequestEvent(), entonces adentro de esta el arduino le dice wire.print(hello) queriendo decir q mande por el cable una cadena string de hello, esta cadena le llega al buffer rx\_tx\_buf, y luego el galileo lo va a imprimir.

En I2C no podemos usar cualquier pin para hacer la conexión, solo en las patas del I2C.

Siempre por estándar, utiliza dos pines del galileo y se conecta a los A4 y A5 del arduino, por estándar, y no pods tener dos masters en un galileo, uno solo también por estándar.

Si queremos conectar otro slave, lo que tenemos que hacer es conectar a una protoboard, utilizar el mismo i2c, y hacer variar el address cada vez y un read