Advanced Computer Systems – Smart Door for Hearing Impaired

By Ayal Kroub and Lavi Kazar

Background & Purpose

The idea of this project is to create a new feature that assists hearing impaired people to know that someone is knocking on their door without having to "turn in" their medical condition to the outside world, and thus to maintain their privacy.

Technically, a micro-controller chip is going to be installed inside the door, and flash a green light to the inner side of it when a door knock is occurring.

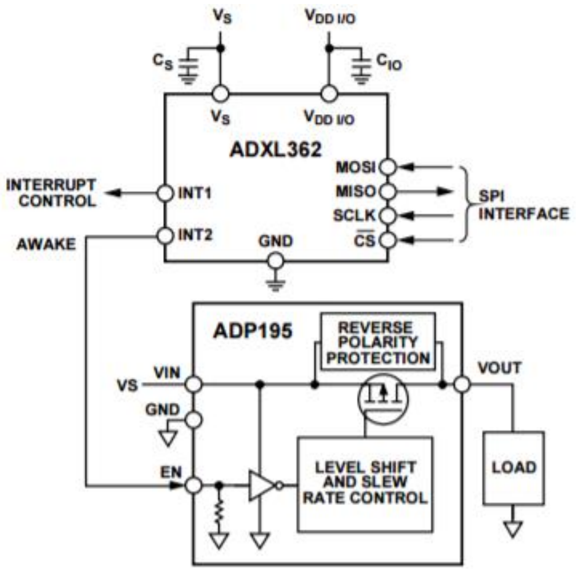
In order to implement this idea we are using TI-RTOS CC2650 micro-controller (MCU), connected to a Sparkfun ADXL362 accelerometer (force sensor). Once installed and set up, the accelerometer constantly listens for knocks and sends the data to the MCU. The MCU analyzes this data and decides whether to indicate a knock or not, ignoring natural noise from the environment.

The communication between the MCU and the sensor is done by SPI protocol, which will be discussed later on in this documentation.

The accelerometer & SPI Protocol

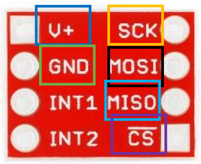
The role of this sensor in our project is to send the acceleration data continuously to the MCU. The data consist of three axis: X axis, Y axis and Z axis, and each ones donation is calculated separately.

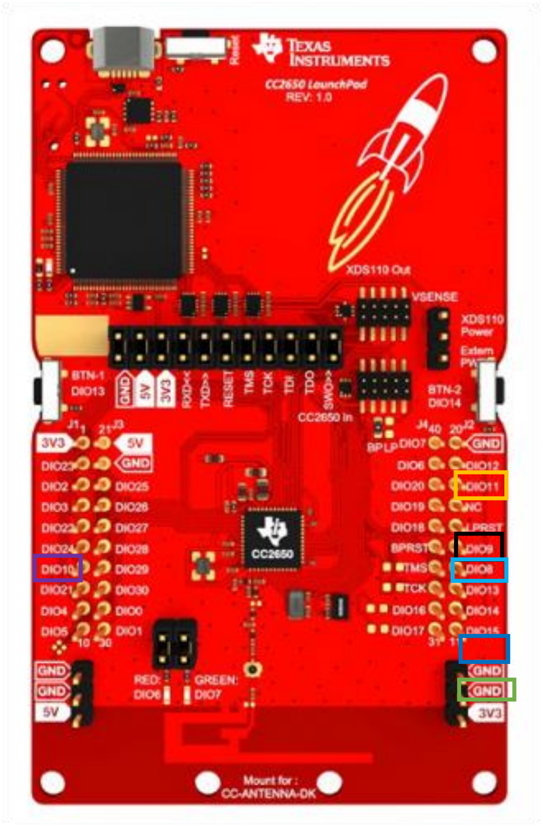
This figure describes the inner block diagram and GPIOs of the sensor side (taken from ADXL362 datasheet):



In order to transfer this data the sensor and the MCU are connected in a certain way that follows the SPI protocol. We connected the sensors MISO to the board according to the formal datasheet, as well as MOSI, CS (chip select), clock and the appropriate voltages.

This figure describes the wire connections between the board and the sensor:

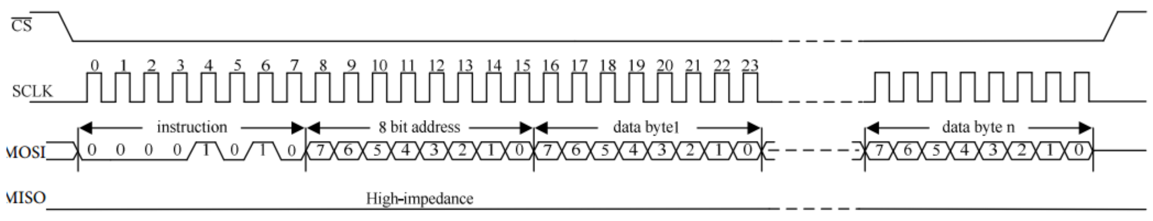




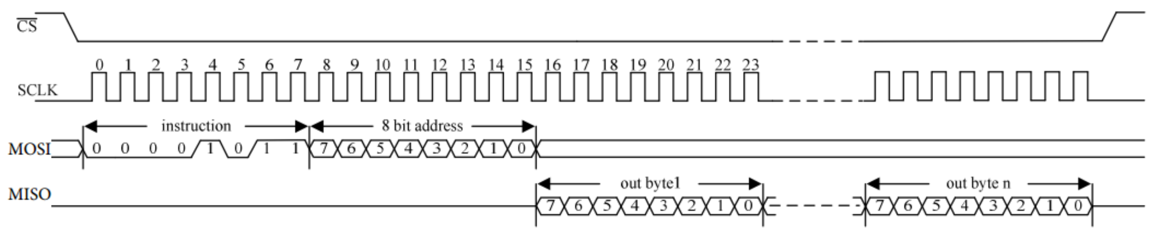
SPI communication is a master-slave communication. In our project, the master is the board and the slave is the accelerometer.

At first, the sensor is in stand-by mode, so a part of its initialization is to change it into Measurement mode by setting the correct bits in the correct register. The SPI protocol works in a permanent flow in which the master samples the slave according to its needs, and fresh data should be available at all times on the slave side. In order to perform the data transfer according to this protocol, the CS value must be low to allow communication. When low, the first byte of the transmitter buffer - in our case the master MCU – stands for the instruction to be executed. The second byte stands for the register to be addressed in the slave, and the third (potentially to the n-th byte) holds the data to be written by the master in case it is going to write some data. After a reading transfer, the receiver buffer will hold the data that the master has read from the slave.

This figure describes the SPI write transaction (taken from ADXL362 datasheet):



This figure describes the SPI read transaction (taken from ADXL362 datasheet):



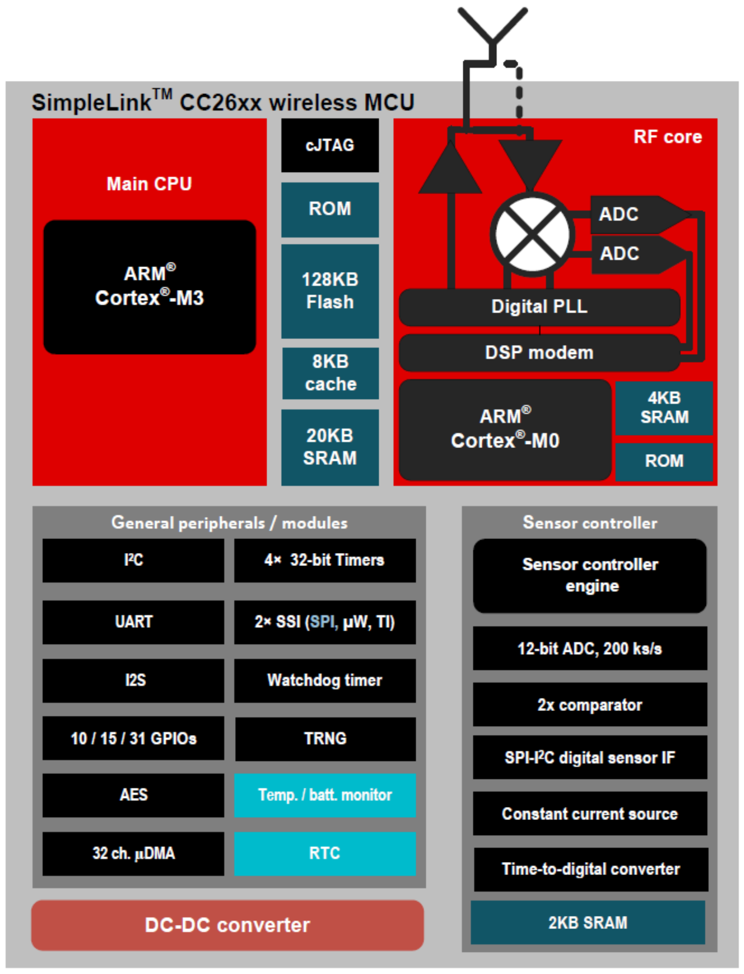
The board & MCU

The CC2650 LaunchPad is a TI evaluation board. The board comes with BLE (Bluetooth Low Energy) capabilities and supports multiple communication protocols, including UART, I2C and SPI.
It also has various built-in sensors and indicators, such as LEDs and a temperature

sensor.
Furthermore, it has a Macronix MX25R8035F flash memory chip of size 8Mb.

We configured most of the SPI parameters to default, with the exception of the clock. We set its frequency to 4MHZ so that it will be coherent with the accelerometer and will get the right data.

This figure describes the block diagram of the board (taken from Texas Instruments datasheet):



Program description

The code that operates this whole program is based on the pinInterrupt project given by TI. We used several libraries that support the board initialization, the sensors header file and standard libc.

Firstly, the initialization of the board and then the SPI communication is made.

Secondly, the SPI task is called and starts setting the configurations for the SPI structures, the communication buffers etc.

The first step in setting up this communication is a certain reset to the sensor which is called ‘self reset’. That too happens through SPI. The second step is to set the sensor to Measurement mode. The last step in our set up is to read the motion register once to define the silent state values.

Then, the program is entering an infinite loop of actively listening to the environment and constantly reading the motion registers. Each time the motion registers are read, the program calculate the difference between the current values and the previous values so it can determine the exact changes of the doors acceleration. Each iteration lasts several milliseconds so it all happens very fast.

Sparkfun allows its clients to measure the acceleration in two ways – a slightly more economic one reading data into 8 bits, and a more accurate one reading data into 12 bits. We chose the latter in light of the frequent noise caused by the environment, especially in apartments buildings.

Finally, in case the program has identified a significant difference between two measurements it calls a method that turns on the green led of the board. We defined the total difference as significant if the difference in all three axis summed up to over 70. The acceleration is measured separately in each axis in values range between 0 – 2^12-1, so the threshold is quite tight. After the LED is turned on the program sleeps for 0.1 seconds and then turns the led back off and sleeps again, and so creating a blinking effect that lasts about 15 seconds.