Challenge 1 – PWM

PwmOut Demo Code

The PwmOut code begins by including the standard mbed library, which refers to the PwmOut library that is used to alter the switching speed of the output to pin to the LED.

The output pin is set to p5 on the mbed and is given the variable ‘led’ for easy reference within the code.

The main code executes an infinite loop that increments the output by 0.1 on a scale from 0 to 1. 0 represents the LED off state, while 1 represents maximum brightness. The program waits for 200ms between every increment and checks if the last increment value is equal to 1. If it is, then the output is rest to 0.

PwmSpeaker Demo Code

Similar to PwmOut, the PwmSpeaker code uses the same PwmOut library to generate a PWM output. In this case, p21 is used as the output for the speaker.

The main code executes and infinite loop to play three different tones that are generated by the function ‘play_tone()’, which accepts the parameters frequency, volume, interval and rest.

- Frequency determines the pitch of the tone.
- Volume determines how loud the tone is.
- Interval determines the duration of the tone
- Rest determines the wait duration after the tone is played.

PWM_Test_LED-NPN

The code begins by including the mbed library and defining a PWM output on p21. A ‘duty_cycle’ variable is set to 0 then the main function begins an infinite loop. The loop increments the duty cycle by 0.1 on a scale from 0 to 1. The period function from the PwmOut object is called to set the output period i.e. the interval between the LED off to full brightness. The ‘write’ function from the PwmOut object is used to set the duty cycle to the LED pin. The program finally waits for one second before the loop is reiterated.

The difference between this program and the previous programs is the use of the ‘write’ function from the PwmOut class rather than directly writing it to the pin. The other difference is the use of the ‘period’ function from the class to set the output interval rather than implementing ‘wait’ delays.

Challenge 2 – SPI

LCD Display Demo Code

The LCD display demo code draws animations on a C12832 LCD display. The C12832 library is included in the code and a C12832 Object is initialized as 'lcd' with the required connections for the SPI MOSI, MISO and Clock.

The animations are bitmaps that are displayed in succession on the LCD, with each bitmap being shifted by a set number of pixels.

Temperature/Humidity Demo Code

This code also includes the C12832 library to display temperature and humidity on the LCD display. The sht31 library is included to communicate with the temperature/humidity sensor over an I2C bus. (Inter Integrated Circuit) The main code executes an infinite loop that clears that first clears the LCD, queries the sht31 device for temperature and humidity values and finally displays that values on the LCD display of SPI. The program check if the temperature has exceeded 25C and illuminates an LED on the mbed if that is the case.

SPI_TEST_MOSIMISO

This code shows an example of using the SPI library for communicating with an SPI slave device directly. The data format and clock frequency of the SPI device is set, and a conditional loop is executed in the main code.

The loop waits for the 's' character from the Serial terminal to exit. In the meantime, the integers of 0 to 9 are written to the MOSI pin and data is read from the MISO pin. A delay is added inbetween each write sequence to allow it to be probed with an oscilloscope.

It can be noted that a chip select was not necessary in this case as since there mbed was communication with a single SPI device and therefore the data would not have caused any interference with other components.

The previous two demos mentioned above make use of existing libraries to simplify the data formatting aspect of the SPI data, whereas this example explicitly defines the format of the data that should be expected by the slave device.

Challenge 3 – I2C

Temperature/Humidity Demo Code

This example makes use of the I2C protocol to interface with the sht31 sensor. The sensor will have a slave address that is pinged by the mbed (master) along with a query. The sensor recognizes its address along with the query and returns a response with its address so that the master device knows where It came from.

I2C_Test_HMC5883L

This code uses the I2C pins on the mbed to communicate with a digital compass. It first gets the address of the device and stores it in an array.

The program then repeatedly requests compass data as long as the user has not pressed ‘s’. The data is then displayed to the serial output. The program makes use of predefined functions in the HMC5883 library to read to and write from the digital compass.

Challenge 4 – Component/Module Selection for MVP

As discussed with the group, components and modules that can be used to build a Minimum Viable Product (MVP) include:

1. HMC6352 Digital Compass
2. EM406 GPS Module
3. SHT31 Temperature and Humidity Sensor
4. RN-42 Bluetooth Module
5. ADXL345 Accelerometer
6. TMP102 Temperature Sensor
7. SRF08 Ultrasonic Rangefinder
8. PC1602F LCD Display