SCC.369: Lab Exercise 3

Communications

Aim of the Exercise
The aim of this exercise is to introduce you to using some of the communications peripherals that are typically found on microcontroller systems. You will also discover how simple serial port communication can be implemented purely in software…

Submission: **SCC.369 moodle submission, 16:00 Friday Week 6.**
Weighting: 25%

The Task

The micro:bit v2 contains an I2C based sensor which is capable of measuring physical acceleration (conveniently called an accelerometer). This device can also sense the gravitational pull of the earth, as a result.

Your task is to write software that **from first principles** reads the three-dimensional accelerometer data from the sensor using the nRF52 TWI module, and reports it via a **bit-banged software implementation** of a 115200 baud serial line to a PC attached vi the USB port.

For this task, we are **not** going to give you all the information you need to complete the task in this document. You will however, find all the information you need in the relevant sections of the datasheet… This will help you practice being an embedded engineer. 😊

Subtask 1: Bit-Bang Serial
Function prototype: `void bitBangSerial(char *str);`

The title says it all!

Write a function **from first principles**, that uses software to bit-bang a 115200 baud TTL serial port implementation that matches the above function prototype. . Once again, **do not include a main() function in your submission** (although you will of course need to create one for your own testing). We will be testing these via an automated test harness, so be sure to use the specified function prototype Your implementation should be stored in a file called Task3.cpp and should:

- Use GPIO to bit-bang the transmit the given null-terminated character string out of the GPIO pin associated with the micro:bit's USB serial port.
- The function should return once all the data has been sent.
- You **do not** need to write a TTL serial receiver – just this transmitter.

- You may use either a timer, busy wait or even a little inline assembler to achieve this task should you wish to feel the burn.

HINT: You will almost certainly need to make use of an oscilloscope to help debug your software. Try using an edge connector pin (e.g. P0) for your output whilst debugging, so you can observe what your waveform actually looks like.. 😉

Subtask 2: Reading Accelerometer Data
Function prototype: **void showAccelerometerSample();**

Using the nRF52 TWI peripheral, write some software that reads the current X, Y and Z axis acceleration data from the micro:bit's onboard accelerometer. You should then use your TTL serial implementation from Subtask1 to display this data on an attached PC. You should update your reading about five times per second.

Your output should take the following format, with a new line ("\r\n") between each reading. The numbers should reflect the accelerometer data provided by the sensor, in the range -1024 - 1024. For example:

[X: 0] [Y: 1020] [Z: -4]

The accelerometer on the micro:bit is an ST microelectronics LSM303 AGR. You can find the datasheet here: https://www.st.com/resource/en/datasheet/lsm303agr.pdf

Assessment

Marks will be awarded for each of the tasks completed, the quality of the code produced, and the quality of the code style you use. This work will be assessed through offline marking and a code inspection of your work.

Marking Scheme

Your work will be marked based on the following three categories. Your final grade will be determined based on a weighted mean of the three weight-bearing grades shown in the table below.

Subtask 1: Bit-Bang Serial	40%
Subtask 2: Reading Accelerometer Data	50%
Code Style and Commenting	10%