Exercise Sheet 5 - Project

After implementing and using communication interfaces such as I2C and SPI and writing drivers for different external units such as displays and accelerometers, we are finally going for the micro-controller project, which is aimed to combine your acquired knowledge and your created libraries into a bigger piece of software, not just being created for learning purpose, but also being some-thing you or other people might want to use! This exercise sheet will give you four options for your project, of which you have to choose only one. The scope is to develop and comment your source code (as in the previous exercises, 15 pts.) and to give a final presentation (10 minutes, 5 pts.) on your implementation. The presentation should include a presentation of your program's structure (supplemented e.g., by a flowchart of your program and/or a flowchart of interesting subroutines) **(3 pts.)**, your individual feedback on difficulties as well as a demonstration of your program **(2 pts.)**. You have four weeks to complete the project, which will be graded by 20 points. Detailed information on the extent of the presentation is provided below.

Option 1: Audio Recorder
Implement an audio recorder which can acquire audio signals from a microphone and store them to the flash memory. Moreover, it should be capable of replaying the audio from the flash.

a) Implement a function to record the audio. Use the digital potentiometer `ISL95811` to control the gain of the input stage and sample the data with the MSP430's ADC with a resolution of 8 bit. Store the captured data in the flash memory **(6 pts.)**.

b) To play the audio, read it from the flash and bring it to the analog domain using the MSP430's PWM module in combination with the on-board low-pass filter (between `CON8` and `IC7`) **(6 pts.)**.

c) Provide an on-screen menu which allows you to start both recording and playback, with the joystick being the input device. The menu should allow to record or replay the audio from different slots in your flash memory (i.e. from different addresses). The audio recordings should feature a minimal recording time of 3 seconds **(3 pts.)**.

Some remarks:

- For debugging your audio recording and playback, directly feed the sampled audio values back to the output, so that you directly hear what you record.
- You might need to use a buffer memory in your RAM, storing the data before writing larger packets to the flash.
- Keep in mind to erase some parts of the flash before writing to them (there might be a delay in-between starting the recording in the menu and really recording the data).

Option 2: Video Game
Develop a video game using the display with custom characters, the joystick for control and the audio buzzer on the base board to generate a nice game sound. It might be a jump'n'run game or something else, be creative!

a) Implement the game itself, coordinating parallel control of display, joystick and sound. You can use a timer to coordinate regular tasks. Implement at least two different levels with increasing difficulty **(10 pts.)**.

b) The game should feature a highscore, which is permanently stored on the flash memory **(1 pt.)**.

c) Implement an on-screen menu for starting the game, entering the player's name and accessing the highscore **(4 pts.)**.

Option 3: Sensor Dashboard
Develop a sensor dashboard which displays the values of all sensor devices on the main and extension boards on the serial interface.

a) Implement a routine for accessing the ultrasonic distance sensor, attached to connector `CON7`. Send a small acoustic impulse, count the time until the impulse comes back to your receiver and estimate the distance based to traveling time and the speed of sound. Calibrate the sensor performing measurements and correcting the data according to your findings **(3 pts.)**.

b) Read and store all sensor data on the board **(3 pts.)**, including
   - the distance (ultrasonic sensor)
   - x-, y- and z-acceleration (MMA8451Q)
   - x-, y- and z-position of the joystick
   - the light intensity (read the LDR on the main board with the MSP430's ADC)
   - the potentiometer status (read the poti on the main board with the MSP430's ADC)
   - the temperature (read the NTC on the main board with the MSP430's ADC)
   - the status of all buttons on the base board (PB1 to PB6, you require the shift register to read some of them)

c) When sending the serial command *sensorDashboard* to the microcontroller, show a dashboard which displays all sensor data captured in b) on the terminal. The dashboard should refresh itself every two seconds. In order to accomplish that, use the so-called *VT100* sequences together with *Putty*[1] as your terminal program. Just refresh the positions of the sensor data, so do not print the whole dashboard again every 2 seconds **(6 pts.)**.

d) Having accessed all sensors by the serial interface, develop a small control library for setting the actuators on the board by the serial interface **(3 pts.)**. This involves enabling and disabling the LEDs on the main board, setting the direction of the relay on the main board, as well as printing text to the LCD (including a function to clear this text). For accomplishing this, use serial functions of the structure *[device] [subdevice] [command] [subcommand]*, e.g. *LED blue enable* or *LCD print "Hello World!"*.

[1] `https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html`

Option 4: Your Own Idea
We do not want to limit you to the given options. If you have a good idea for a project using the MSP430 as your core device, feel free to propose it to us. You can also use additional hardware not being found on the main or extension board. Your proposal should roughly have the same scope as the other options (credit-wise, the project is 1/3 of the course, i.e. 1 ECTS, being equivalent to 30 h of workload). To proceed with your own idea, please contact us during the first week of the project's time period, so that we agree on scope and point distribution of the **15 pts.**.

Information on the presentation of the project
You will need to present your project in a recorded video which should be between **5 to 10 minutes** long and needs to be uploaded as a ZIP-File. You will be provided a link by mail on time which will allow you to upload ZIP-Files up to **3 GB**. Please make sure that your video contains

a) a short description of the project with an illustration of your program flow with the most important program routines. You are allowed to use slides to illustrate your description/program flow. We do not need a detailed description of the complete source code.

b) a live-demonstration of all the functionalities of the listed features in the project description on your hardware.

c) a description of at least one special feature of the program with your greatest individual contribution.

You will also need to bear in mind that

a) your uploaded ZIP-File is named after your name and student-number, e.g., John_Smith_49023.zip. Be aware that files, which cannot be assigned to a student, will **not be accounted** in the grading process.

b) your uploaded ZIP-File includes **both the video and the source code files** and **avoid** uploading each file separately.

c) you stick to the time limit of **10 minutes** maximum. We reserve the right to deduct points if the video is too long.

d) your video shows **all** functionalities mentioned in the project description. We reserve the right to deduct points from both the presentation part (5 pts.) and implementation part (15 pts.) otherwise.

e) the quality of the presentation is **adequate**. A combination of showing slides and the implementation on the board (simultaneously) might be a good way of presenting the project. However, you are completely free on the way of presenting your project as long as you keep the aspects mentioned above in mind.