Ruzica Piskac, Ph.D.
Leander Tentrup, M.Sc.
Michael Gerke, M.Sc.
Felix Klein, B.Sc.

Summer term 2014
Project 3
Due: Friday, 25th July 2014

Embedded Systems

In this project we aim to replicate the mars pathfinder robot, sent to mars in 1996. However, as the complete robot would be out of scope of the lecture we reduce the setting to a slightly simplified version. The goal of this project is to illustrate the design workflow, the interplay of different components in a larger embedded device, and handling access to restricted resources. Furthermore, using a hardware description language to design time critical components illustrates the main conceptual differences compared to usual programmable microcontrollers.

Introduction. In our setting, we consider three main components of the robot: the meteorological data gathering task (M) collects data from its environment and writes it on the information bus, the bus manager (B) controls access to the bus and transports data over the bus to its destination, and the communication task (C) exchanges data with the base station. To represent M and C we use the Arduino Nano v3. The bus manager is controlled through a Spartan6-LX45 FPGA. The Arduino (M) is equipped with an acceleration sensor providing the "meteorological" data to the system. The Arduino (C) is equipped with three LED bars to display the meteorological data. Furthermore, a button is used to model a request from the base station. Additionally, all three components are equipped with an IR-receiver and an IR-transmitter.

The behavioral description of the project can be summarized as follows: Repeatedly, M collects the three values from the acceleration sensor and sends it to the bus via the IR-transmitter. B then collects this data and saves it until new data arrives. Finally, if C requests data (triggered by a button press at input pin 12) B forwards the last received measurement to C. To ensure mutual exclusive access to the bus, B has to schedule M and C using a channel access protocol in which C has priority against M. Details to the different components and protocols are given in the next sections.

Meteorological Data. We use the acceleration sensor MMA7361[1] to collect acceleration values on all three axes x, y and z. To access the sensor use the library at:

`https://code.google.com/p/mma7361-library/`

Note, that we do not use the example wiring on the board. Use the following code to initialize the sensor:

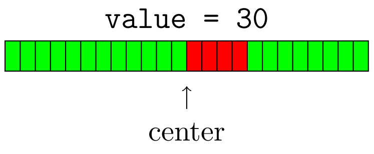
```
<acc>.begin(5,8,6,7,A0,A1,A2);
<acc>.setARefVoltage(3.3);
<acc>.setSensitivity(LOW);
<acc>.calibrate();
```

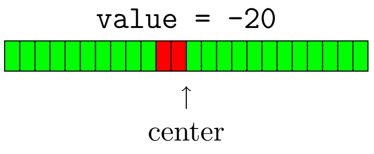
Here, `<acc>` represents the variable to your sensor interface.

Displaying Data. We use three bi-color LED bargraphs[2] equipped with 24 red and green LEDs each to display the received acceleration data. Use the library at

`https://github.com/adafruit/Adafruit-LED-Backpack-Library`

to control the bars. Each bar is used to display one axis of the received acceleration data. We use an amplitude to display positive and negative values where the center represents zero. For a positive value, the amplitude should go to the right and for a negative value the amplitude should go to the left. The amplitude is displayed by enabling the red LEDs. If there is no amplitude, the green LEDs should be enabled. Consider also the following illustrations of two examples.





Data Transmission. We use the infrared LEDs LD 271 L[3] (output pin 3) to transmit data and the infrared receiver TSOP4838[4] (input pin 2) to receive data. To visualize the transmissions each component is additionally equipped with a LED. To transmit the data on the Arduinos use the library at:

`https://github.com/shirriff/Arduino-IRremote`

[1] `https://www.sparkfun.com/datasheets/Components/General/MMA7361L.pdf`
[2] `https://www.adafruit.com/products/1721`
[3] `ftp://ftp.dca.fee.unicamp.br/pub/docs/ea079/datasheet/ld271.pdf`
[4] `www.vishay.com/docs/82459/tsop48.pdf`

Due to the current consumption and to visualize the IR-signal we use two PNP-transistors which act as a inverter. Correspondingly, you have to change the following lines in `IRremoteInt.h` and `IRremote.cpp` of the library:

```
IRremoteInt.h, line 204:  #define MARK 0 → #define MARK 1
IRremoteInt.h, line 205:  #define SPACE 1 → #define SPACE 0
IRremote.cpp, line 282:   digitalWrite(TIMER_PWM_PIN, LOW); →
                          digitalWrite(TIMER_PWM_PIN, HIGH);
```

Due to its simplicity, we use the IR-remote protocol developed by NEC to encode our data. This protocol can be used to transmit four byte data values (32 bit). Detailed information about the protocol can be found at:

`http://www.sbprojects.com/knowledge/ir/nec.php`

This protocol is already implemented in the IR-Remote-Library. Note that we do not use message verification.

Bus Management. Our bus is controlled by a Spartan6-LX45 FPGA which is embedded on a Digilent Anvyl^{TM} Spartan-6 FPGA Development Board[5]. For the project you have to complete the given project files, which are available on our webpage. Your task is to implement

a) the decoder (decoder.vhd) which is used to parse the IR-protocol from NEC and

b) the bus manager (manager.vhd) which schedules M and C, saves data and reads/writes data to the bus.

To ensure mutual exclusion on the bus, you should use the following protocol:

B	**M**	**C**	**B**		**B/M**	
ROUND_SYNC	ACCESS_REC	ACCESS_REC	ACCESS_M ACCESS_C	ACC_X	ACC_Y	ACC_Z
t	t	t	t		$3 \cdot t$	

The protocol is organized in rounds. First, B sends out a message `ROUND_SYNC` to synchronize M and C to the protocol. Then M sends out the message `ACCESS_REC` if it has new data. The same follows for C if the button was pressed. Afterwards, B either sends out `ACCESS_M` or `ACCESS_C` depending on which component has requested the bus. Remember that C has priority over M. Finally, either B sends the three data values to C or B receives new data from M. An appropriate value for t is $t = 200ms$, but you may optimize this. After the end of a round a new round starts.

You have to implement the scheduling protocol on all three devices. You should make sure that there no interference between the receiver and transmitter of the same component. Furthermore, you have to find appropriate four byte values for the messages `ROUND_SYNC`, `ACCESS_REC`, `ACCESS_M` and `ACCESS_C`. You can use the segment display and LEDs for debugging purposes.

[5] `https://digilentinc.com/Products/Detail.cfm?NavPath=2,400,1175&Prod=ANVYL`

In your final solution

1. you should only make changes to the marked areas,
2. your project must compile without producing any warnings/errors, and
3. the reset signal should be handled correctly.

Hardware. All required hardware can be found in E1.3 Room 301. Please do not move the FPGA, you just need the USB cable and the power switch. Do not change the circuits as they are pre-configured. Take care that all devices are powered off, if you leave the room.

Problem 1: Sub-Goals

Try to archive the following sub-goals first:

1. Install all mentioned Arduino libraries, apply the necessary changes and get familiar with the interfaces.
2. Install the Xilinx ISE Design Suite (free WebPACK license) from `www.xilinx.com` (a registration is required)[6]. After the installation you need to install the Digilent driver which can be found under

   ```
   <path to your installation>/ISE_DS/ISE/bin/<os type>/digilent/
   ```

   As there might occur problems when downloading the Linux binaries, you can also get them directly at our offices.
3. Get familiar with the Design Suite interface and the simulator.
4. Try to understand the already provided project files and their design concepts.
5. Try to send measurements directly between M and C.
6. Try to receive messages by the FPGA. You can use the decoder to test some cases with the simulator.

It may be useful to parallelize some of these sub-goals.

Problem 2: Final Goal

Complete the project and send the files `Arduino_m.ino`, `Arduino_c.ino`, `decoder.vhd` and `manager.vhd` before 25.07.2014 23:59:59 to

`es14@react.uni-saarland.de`

Add your group number, your names and your immatriculation numbers to the e-mail. Be prepared to present your solution eventually after the exam.

[6]In case you use OS X, you have to set up a Linux or Windows VirtualBox first, as there is no direct support for this system by Xilinx.