Graz Technical University

Institute for Technical Informatics

Computer Systems and Networks UE, WS2019/20
Konrad Diwold
kdiwold@tugraz.at

Assignment 1: Bare Metal Raspberry Pi Setup (5pt)

Deadline: October 24, 2019; 23:59 CET

In this assignment, you will set up your local programming environment for "bare metal" development on the Raspberry Pi platform and create, compile, and run a few simple programs on the hardware.

(1) Toolchain Setup (0.5pt)

In this task, you will setup and test your toolchain for compiling and deploying bare metal programs onto your Raspberry Pi.

(a) **Setting up your Programming & Compilation Environment** Set up a GCC ARM compilation environment on your machine (available for Linux, Windows and Mac from here2), we will be working with the **arm-none-eabi-xx toolchain** for 32bit systems (i.e. ARM processor / NO operating system / embedded Application Binary Interface). Use the latest version of the toolchain. After setting up the toolchain, ideally add the /bin folder of the installed software to your system's PATH[1] and make sure that the commands *arm-none-eabi-gcc* and *arm-none-eabi-objcopy* work as expected from a console window (e.g., in Windows, the Command Prompt cmd). Follow the tutorial and try to compile the following dummy program:

```
int main(void) { while(1); return 0; }
void exit(int code) { while(1); }
```

(b) **Getting the Raspberry Pi Ready for Bare-Metal Programming**
Use Etcher (https://etcher.io/) to flash your SD card with Raspbian Stretch Lite (from https://www.raspberrypi.org/downloads/raspbian/), a Debian version specifically designed for Raspberry Pis. This will place the Raspbian's bootloader on the card – we recommend this to enable you to easily switch between the full Raspbian operating system and your own code (that does not require an operating system) in the future: to do this, you can now simply replace Raspbian's operating system kernel (`kernel.img`) with your own code (see next task) – make sure that you keep the original `kernel.img` as well, to be able to switch back (by renaming the files).

(c) **Testing your Toolchain**
Compile the program and put it on the Raspberry Pi. Nothing should happen...

(2) Bare Metal "Hello World" (4.5pt)

In this task, you will create a digital circuit and program that will blink an LED – a classic "Hello World" program for embedded systems.

(a) **Connecting all Components**
Use your cables, resistors and breadboard to properly connect an LED to your Raspberry Pi. Test the setup using the 3V power and ground pin.

[1] There is lots of instructions on how to do this online...

(b) **Bare Metal ACT LED Blinking (1.5pt.)**

Follow the tutorial at http://www.valvers.com/open-software/raspberry-pi/step01-bare-metal-programming-in-cpt1/ to build a piece of software that blinks the Pi's Status/ACT LED 3 times, then waits for a short time before repeating. When transferring that program to the Pi, first rename the existing Raspbian `kernel.img` and then place your compiled and extracted `kernel.img` on the SD card.

For this exercise, in particular make sure that you understand where the GPIO register indices and positions in memory required for controlling the Status LED come from and how they are calculated (the tutorial references https://www.raspberrypi.org/app/uploads/2012/10/Raspberry-Pi-R2.0-Schematics-Issue2.2_027.pdf and https://www.raspberrypi.org/app/uploads/2012/02/BCM2835-ARM-Peripherals.pdf and https://www.instructables.com/id/Bare-Metal-Raspberry-Pi-3Blinking-LED/ the main sources for this information)

(c) **Bare Metal ACT LED Blinking (2pt.)**

Connect your led setup to the raspberry GPIO Pin 24. Modify your code to blink the external LED that you connected to GPIO 24 instead of the internal ACT LED. To solve this puzzle, you will need to consider again the BCM2835's manual and the webpages mentioned above.

(d) **Understanding the BCM2835 (1pt.)**

Given you read the tutorial and know how to control access the leds at the various registers, please outline and explain in your report

1.) What is `gpio[LED_GPFSEL]` (in the code that blinks the `ACT LED`)?

2.) How would you modify this line in order to turn all `GPIO` pins that are covered by the GP Function Select Register 2 into outputs?

(3) BONUS: LED Experiments (0.5pt)

Extend your program with functionality that fades the LED in slowly instead of just turning it on. There are multiple options of how this can be accomplished – any functionally correct way will be accepted as a solution to this task!

(4) Hand-in Instructions

By the deadline, hand in via StudyNet: a **zipfile** that contains (i) your **code** and (ii) a **short report** that states the team members of your team and their roles when doing the assignment, pitfalls you encountered while working on the assignment, and instructions about how to run your code (if non-obvious).