Project report: Internet of Things

Introduction

1 Setup

1.1 Networking

During the development and the testing of the project, I used an Ethernet cable with a static networking configuration.

The Raspberry Pi, my home computer and the Internet connection were all connected to a Dlink switch.

In order to define a non-volatile network configuration on the Raspberry Pi, the file `/etc/network/interfaces` was edited by adding the following instructions:

```
iface eth0 inet static
  address 192.168.1.11
  netmask 255.255.255.0
  gateway 192.168.1.1
```

My Internet connection is backed by an Orange Livebox which is on a 192.168.1.0/24 network. It supports DHCP but I preferred to give the Raspberry Pi a static configuration in order to ease the configuration of the SSH link between the devices.

1.2 SSH

SSH is a secure transport protocol designed to replace Telnet. It can be used to send commands to a remote device or setup things such as port forwarding, encrypted tunnels, etc. I have decided to setup SSH on the Raspberry Pi in order to drop the need to use the QWERTY keyboard, the mouse and another screen.

The SSH server service has been enabled on the Raspberry Pi, meaning it will start during the boot. This has been achieved with this command:

```
sudo systemctl enable ssh
```

After that, I created a new SSH key on my home computer without a passphrase using `ssh-keygen` with its default options. The key was then sent to the Raspberry Pi using:

```
ssh-copy-id -i ~/.ssh/rpi pi@192.168.1.11
```

To make things easier, I also defined a SSH alias in `~/.ssh/config`:

```
Host rpi
  IdentityFile ~/.ssh/rpi
  User pi
  HostName 192.168.1.11
```

This enabled me to connect to the Raspberry Pi under the alias `rpi` and without providing a password. From now I can type directly `ssh rpi` or use `rpi` in the `rsync` or `scp` commands.

1.3 Hardware

1.3.1 The basics

1. Open the Raspberry Pi in order to access the GPIO pins.
2. Connect the GPIO "rainbow pride" cable to the Raspberry Pi.
3. Connect the GPIO extension board to the other end of the GPIO cable.
4. Plug the GPIO extension board into the middle of the breadboard.
5. Plug a redcable between the `5V0` pin and the `+` column on the right.
6. Plug a blackcable between the `5V0` pin and the `-` column on the right.
7. Plug a redcable between the `3V3` pin and the `+` column on the left.
8. Plug a blackcable between the `GND` pin and the `-` column on the left.

1.3.2 RGB LED module

1. Plug a cable between the `VCC` pin and the `3V3 +` column on the left.
2. Plug the `R` pin of the module into the `GPIO17` pin of the GPIO extension board.
3. Plug the `G` pin of the module into the `GPIO18` pin of the GPIO extension board.
4. Plug the `B` pin of the module into the `GPIO27` pin of the GPIO extension board.

1.3.3 PCF8591 module

1. Plug the `VCC` pin of the module into the `3V3 +` column on the left.
2. Plug the `GND` pin of the module into the `3V3 -` column on the left.
3. Plug the `SCL` pin of the module into the `SCL1` pin of the GPIO extension board.
4. Plug the `SDA` pin of the module into the `SCL1` pin of the GPIO extension board.

1.3.4 Thermistor

1. Plug the `VCC` pin of the module into the `3V3 +` column on the left.
2. Plug the `GND` pin of the module into the `3V3 -` column on the left.
3. Plug the `SIG` pin of the module into the breadboard, on a line where it is not connected to the extension board. On the same line, plug a cable and connect the other end to the `AIN0` pin of the PCF8591.

1.4 Software

First, we need to update the package cache in order to be able to install new packages:

```
sudo apt-get update
```

1.4.1 The Go toolchain

Then, we will need the Go toolchain for local testing. The package available in Raspbian is too old, so here is the command for the last release:

```
wget https://dl.google.com/go/go1.10.linux-armv6l.tar.gz
sudo tar -C /usr/local -xzf go1.10.linux-armv6l.tar.gz
echo "export PATH=$PATH:/usr/local/go/bin" >> ~/.bashrc
```

After that, we need to setup Go environment in a specific folder and permanently export the `GOPATH` environment variable:

```
sudo su -
mkdir -p ~/go
echo "export GOPATH=~/go" >> ~/.bashrc
source ~/.bashrc
```

Note we are doing this with the `root` user because only the superuser can play with PWM (used by the RGB LED).

1.4.2 Cloning the repository

Git is already installed on the Raspberry Pi. We will use it to clone the repository of my project hosted on GitHub.

Note that the repository must be cloned in `$GOPATH` in order to be able to use `go get`:

```
sudo su -
export IOTDIR=$GOPATH/src/github.com/efournival
mkdir -p $IOTDIR && cd $IOTDIR
git clone https://github.com/efournival/iot-project.git
cd iot-project
```

1.4.3 Enabling the I2C bus

The I2C bus is used by the PCF8591 analog-to-digital converter in order to read the temperature from the thermistor.

First, connect to the device using SSH (for example, `ssh rpi`) and run the configuration utility:

```
sudo raspi-config
```

Then:

1. Enter `9 Advanced Options`.
2. Enter `A6 I2C`.
3. Answer “yes” to the question “Would you like the ARM I2C interface to be enabled?”.
4. Wait for the confirmation then quit with the escape key.

The device driver should now be available under `/dev`, you can quickly check by running `ls /dev/i2c-1`.

Conclusion