University of Cape Coast

Department of Computer Science and Information Technology

Embedded systems Quiz 1

Course Code: CSC321 Duration: 1hr 20mins Date: 16th February 16, 2023

The DHT11 Temperature and Humidity Sensor

The DHT11 is a digital temperature and humidity sensor featuring its own proprietary communication protocol. It uses a single GPIO line (GPIO 22) to serially transfer its temperature and humidity measurement.

MicroPython already has a driver for the DHT type sensors (DHT11 and DHT22) included in its binary making it very easy to read the temperature and humidity values. The driver however hides the protocol from the user and the user is unable to see what is going on behind the scene.

For this reason, a second driver allowing to read raw bit values as they appear in the DHT11 protocol has been added to the ESP32. This will help you understand how the DHT11 actually works and what the driver is doing to extract the temperature and humidity values.

This driver (module dht11Raw) starts a measurement on the DHT11 and then scans the DHT11's data line every 4 us keeping it state in a 32*32bit array. The first measured bit is the highest significant bit of the first array element and the last measured bit is the lowest significant bit of the last element in the array.

The DHT11 data sheet has been attached to this document.

Question: Write a script **dht11.py** which extracts data from the DHT11 automatically and prints it every 30s.

A guide is provided below

Getting the Raw Data

Get the raw data of the DHT11 and write it to a file on the ESP32. The *dht11Raw* module has a single function: dht11ReadRaw (pin, dht11Data)

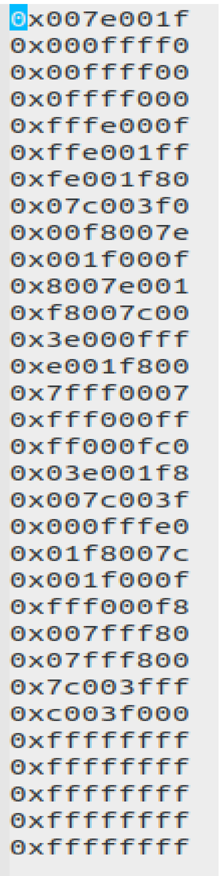
The data must be an integer array with 32 integer values. You can create this with:

import array
*dht11Data = array. array("I",[0]*32)*

This will create the needed array and fills it with zeros. Once you have the data,
write the data in a suitable format to a file on the ESP32 with the path "/data/dht11.txt".
Before writing to the file, check if the "/**data**" directory exists. Make use of **try/except** clause using the **system call** (uos) to validate the existence of "/**data**".

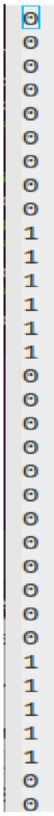
Write the data in hex text format with 8 hex characters per integer number.

Example:



```
0x007e001f
0x000ffff0
0x00ffff00
0x0ffff000
0xfffe000f
0xffe001ff
0xfe001f80
0x07c003f0
0x00f8007e
0x001f000f
0x8007e001
0xf8007c00
0x3e000fff
0xe001f800
0x7fff0007
0xfff000ff
0xff000fc0
0x03e001f8
0x007c003f
0x000fffe0
0x01f8007c
0x001f000f
0xfff000f8
0x007fff80
0x07fff800
0x7c003fff
0xc003f000
0xffffffff
0xffffffff
0xffffffff
0xffffffff
0xffffffff
```

Extract the individual bits from the raw data and write them to an "uncompressed file". Each data entry will contain either 0 or 1 (convert the hex values to Binary). Example below



```
0
0
0
0
0
0
0
0
0
1
1
1
1
1
1
0
0
0
0
0
0
0
0
0
0
0
0
1
1
1
1
1
0
0
```

Read the raw data and **Extract temperature** and **humidity** from this measurement and finally calculate the **checksum.**