SUSIE Report

version 1

Johannes Gårdsted Jørgensen - s093457

May 11, 2015

Contents

Nabovarme

Introduction

A vocabulary of technical terms used in this report is found in the appendices.

This report describes a part of the process of digitalizing the heating infrastructure at Christiania, Copenhagen. The heating infrastructure goes under the name "Nabovarme" and is organized as a collection of wooden pellet stoves heating groups of households or apartments through heated water and radiators. Nabovarme currently uses pen and paper to document each customers heating usage by checking the heating meters in each household once a year. It was decided that digitalizing the collection of usage data would give better data and data with a much higher frequency. This higher frequency would first of all make the billing much easier since all data was allready existing in the database when billing time came around. The higher frequency however would also open up a list of oportunities like giving customers the ability to follow, and react upon, their usage and maybe lead to a lower pellet usage for whole Christiania. I started this project as a prototype that includes a device that transmits household heating usage to a central server that will visualize it in html.

Problem formulation

Nabovarme wants a device that is able to communicate with the class of heating meters found in all households in Christiania. The device has to be embedded inside the heating meters and use the same builtin power supply if possible. The device has to sample at a predefined frequency and sleep between samples if possible, all samples must have been sent a central server or at least attempted to be sent once. The device has to use wifi to transmit samples and an NTP (network time protocol) client must exist on the device to timestamp all samples. The device has to be easily configured, with regards to what wifi hot-spot it will use for transmission, and it has to strive for end to end encrypted packages so interception is not an issue. The device has to be cheap and relatively low powered and available in large quantities. A pcb will have to be made for connecting the device to the heating meter.

Requirements

Name	Description
R1	The protocol for serial communication with "kamstrup multical 601" has to be reverse engineered and implemented in c.
R2	The device must use the protocol MQTT to transmit samples to a central server
R3	The device must use wifi for data transmission and feature a grace period for configuring which accesspoint to join
R4	The device must use dns for looking up the ip of the central server and use ntp for getting the unix timestamp it uses in its RTC.
R5	A central server has to receive and save the samples in a sql database and the same server has to serve a demo web page where the data of a single device can be seen visualized as a graph
R6	The device has to sleep between samples
R7	The device has to to transmit all samples at least once if possible (QOS)
R8	A pcb should be designed and manufactored for the device to be placed inside the kamstrup meter

Design

The system can be described using the diagram beneath:

TODO: image of whole system

The heart of the system is the Kamstrup Multical 601 meter. This meter is found in households in Denmark and other countries as well. The Meter is normally the property of the company providing the heating infrastructure. Therefore they are mostly locked to keep endusers from mangling and manipulating the device. This however is not the case at Christiania where we have bought the meters and installed them ourself. This gives us a rare opportunity to open the meter and install new electronics inside it where Kamsturp left room for expansion boards like shown underneath:

TODO: image of open kamstrup meter

Connected to the kamstrup meter is a microcontroller using a UART connection to talk to the Kamstrup meter. The microcontroller has onboard wifi and and external antenna is connected to the device with an extension cable. Both the microcontroller and the Kamstrup meter gets power from a builtin power supply.

The "meter unit" (Kamstrup meter and wifi microcontroller) has an established connection to a nearby wifi hot-spot. The meter unit transmits samples through the wifi hot-spot, over the internet to a central server.

The central server collects samples and serves a web page where the samples can be seen in a graph.

Implementation

Hardware

The hardware in this project is pretty generic except of the embedded device in the meter. The server can be any linux server with the ability to run an ntp server, web server, a mysql database and a MQTT broker. The wifi hot-spots can be of any kind and is not configured by Nabovarme.

The embedded device was chosen based on these features:

- Price
- Onboard wifi
- Power consumption
- Onboard flash
- Clock speed
- Ease of development

The following embedded solutions were explored:

- Arduino with cc3000 wifi modem
- Raspberry pi with usb wifi modem
- Esp8266 microcontroller (RISC)

Based on the research i created the following table:

Solution	Price	Onboard wifi	Transmit Power consumption	Onboard Flash	Clock speed	Development
Arduino with cc 3000	ca. 75$ (40 + 35)	No	ca. 270mA, (20 + 250)	16kb	16Mhz	C like, easy
RaspberryPi, edimax usb wifi	77 (52 + 25)	No	250mA (210 + 40)	None	700Mhz	Linux, easy
Esp8266	2,5 $	Yes	215mA	512kb	80Mhz	C sdk, moderate

sources: http://www.arduino.cc/en/Main/ArduinoBoardDuemilanove avr atmega 328 datasheet cc3000 datasheet https://nurdspace.nl/ESP8266 http://raspi.tv/2014/pihut-wifi-dongle-vs-edimax-power-usage https://github.com/esp8266/esp8266-wiki/wiki/W25Q40BVNIG

I concluded from the data above that the ESP8266 with a price of 2.5$, onboard flash and 80 mhz clock speed was ideal as a platform for the project.

ESP8266

The ESP8266 is a new microcontroller with onboard wifi and 512 kb spi flash. It was introduced to the global market by Espressif in 2014 and has since gained a large following in hacker and maker circles around the world.

Espressif has released an sdk for the ESP8266 and have kept updating the sdk for the bimonthly. The sdk has some binary blobs but a project called Esp-open-sdk tries to replace these with open source alternatives. The binary blobs make it a little dificult to develop software for the esp since you cant research the inner workings of specific core api calls.

The sdk is based on freertos and give you a task handler asn other abstract os features. Most of its core api calls are callback based which means you have to think asynchronously when developing.

Software

Software had to be developed to extract the samples from the Kamstrup meter and send them to a central server In order to do this a few core protocols must be known.

Protocols

I have chosen to document the kamstrup multical 601 protocol since its not available in the public domain and protected by an NDA so i regard the work done on freeing the protocol as a public service. The MQTT protocol is also documented more loosely since its freely available. DNS and ntp are not documented since they not represent major parts of the system and are documented many places.

Kamstrup multical 601

df

MQTT

asf

Esp8266 firmware

her is an unfinished introduction

DNS, NTP, RTC and sleep

here is some text describing what and some codee snippets showing how it was implemented

```
void ICACHE_FLASH_ATTR
ntp_udpclient_recv(void *arg, char *pdata, unsigned short len) {
    struct tm *dt;
    char timestr[11];
    // this is NTP time (seconds since Jan 1 1900):
    uint32 timestamp = pdata[40] << 24 | pdata[41] << 16 |
        pdata[42] << 8 | pdata[43];                  // BUG? is ntp 32 or 64 bit?
    timestamp =  timestamp - NTP_OFFSET;
    set_unix_time(timestamp);
}
```

Kamstrup

here is some text describing what and some codee snippets showing how it was implemented

MQTT

here is some text describing what and some codee snippets showing how it was implemented

Analysis

Conclusion

this is a conclusion

Appendices