control and monitoring of a solar thermal system

based on Arduino Uno and Raspberry Pi

Kevin Bundschuh
Ubiquitous Computing Laboratory, Department of Computer Science
University of applied Science (HTWG) Konstanz
Konstanz, Germany
kebundsc@htwg-konstanz.de

***Abstract*—During the last thirty years the share of renewable energies (water power, wind power, solar power, …) in the German-energy-mix increased by a factor of ten [1]. Being part of this development with the own house requires a bearable investment in one of the mentioned technologies, but only in the case of building a new house with no existing heating system for water and space. The integration into an existing heating system could be complicated and even more expensive, because most systems available on the market are not flexible enough to fit in the given requirements. The goal of this project is to set up a control unit (based on an Arduino Uno) and a monitoring system (based on a Raspberry PI) for a solar thermal system, which was integrated into an existing heating system. The paper focuses on two main topics. The first topic is about gathering several sensor data processing them and depending on the results controlling valves and pumps. The second topic mainly deals with the visualization of the gathered sensor data on mobile devices.**

Keywords—solar thermal; control unit; monitoring system; node RED; amqp;

I. INTRODUCTION

Taking care of the natural environment became a very important part in many people's lives over the last few years. In Germany there is a big trend in using renewable energies for heating freshwater and space in private homes. But not only the environment aspect forces the growing use of these technologies since fossil power carries like oil or coal became more and more expensive over the last decades.

The two most suitable approaches for using renewable energies in private homes for heating are heat pumps using the earth's ground energy and solar thermal systems using the power of the sun. The following paragraphs explain the basic working principle and mention the main advantages and disadvantages.

A. Heat pumps

Heat pumps are using the temperature energy stored in the ground. Only a few meters below surface the ground´s temperature is never below a certain value depending on regional conditions and how deep hole is drilled. The basic idea is to find a carrier, which gets gaseous above this temperature value and fluid if its below. The carrier fluid is now pumped through the pipe in the drill hole afterwards the gaseous carrier will be compressed by the so-called heat pump. Compressing gases generates heat, that can be used for heating.

The main advantage of this technology is, that it is endless available, there is no need to store the energy. One disadvantage is the need of additional electricity to run the heat pump. Another disadvantage is, that drilling the hole could be expensive compared to other technologies.

B. Solar thermal systems

Solar thermal systems are using the sun's energy for heating up a carrier fluid. Heating the fluid is done in the so-called collectors, which are normally placed on the roof. After that the fluid is being pumped through a heat exchanger where the carrier is cooled down[2].

These systems are cheaper than most others and are very easy to understand and maintain. But the sun as energy source is not always available. This fact requires to store the energy. Typically, water is used for storing. The water is heated up in the heat exchanger and afterwards stored in big isolated tanks.

II. MOTIVATION

The goal of this research is to develop a low-cost control unit, that controls several pumps and valves for a solar thermal system, which was integrated in an existing heating system, that is reliable and fulfill basic safety and security criteria. Further there should be a graphical user interface to monitor all necessary sensor data in real time, which is accessible over the internet by entitled users. The control logic must not involve a high computational effort, in order to run on embedded platforms like the Arduino Uno board. Sensors for temperature are considered, since they are available in various configurations and easy to implement. Internet of Things (IoT) and Arduino compatible frameworks and platforms are preferred for fast development of a low-cost, energy efficient and Internet connected solution. This paper reports on the progress towards achieving the goal. It will describe in detail the working principle of the control unit and the graphical interface.

III. State of the art

Currently, the gold standard in terms of solar system regulation and control is the comparison between the two temperature values: The temperature at the collector and the temperature in the freshwater tank. There are several products on the market, which consider the usage of additional buffer tanks, but they only have a few options to select how the different tanks are combined in an existing heating system.

There are two approaches for standard regulation. The first approach is the so-called "Hight-Flow-regulation" the regulation strategy aims to keep the collector temperature as close as possible at the lowest temperature in the tank. The actual flow of the carrier fluid is here about 30 to 50 liters per m² collector face. The second approach is called "Low-Flow-regulation", in this case only a certain part (upper part in most cases) is heated quickly[3]. The carrier fluid here is about 10 to 20 liters per m² collector face. This saves the power consumption of the collector pump.

Defining a current standard for implementing graphical user interface, which is available over the internet is very hard, since there are many options to realize. One option could be developing a state of the art cross platform application and combine it with a real time database such as the google firebase. Another option is setting up a webserver running a webpage. The webpage would also connect to a database where the temperature values are stored. A suitable option could also be using the Node-Red framework, which allows the easy usage of IoT protocol interfaces like MQTT or AMQP to transfer sensor data in real time, so no additional database is needed therefore anymore. Further this framework offers to setup a dashboard with gauges and graphs for visualizing the data.

IV. Control unit

The existing heating system consists of two water storages with 1000 liters each. These storages are called buffer and used for space heating. There is also a 200 liters tank for fresh water, used for drinking, showering, …. The fresh water tank and buffers are heated up by a wood heating, that must be filled up and started manually almost every day in winter times.

The additional solar system is now supposed to support the wood heating by heating up the buffers and fresh water with the power of the sun. This should now safe work and wood especially during the summer, autumn and spring months.

Figure 1 shows the integration of the solar system into the existing heating system. Inside the collector the carrier fluid is heated up. The collector pump circulates the carrier fluid within the system. The collector valve decides whether the fresh water tank or one of the both buffers is heated. If buffer one or buffer two is heated the carrier fluid runs through an external heat exchanger. In case of heating fresh water there is no external heat exchanger needed because the fresh water tank has one integrated. The water stored in the buffer tanks is circulated by the buffer pump. Valves two and three decide whether the water inside buffer one, buffer two or none of both circulates trough the heat exchanger.

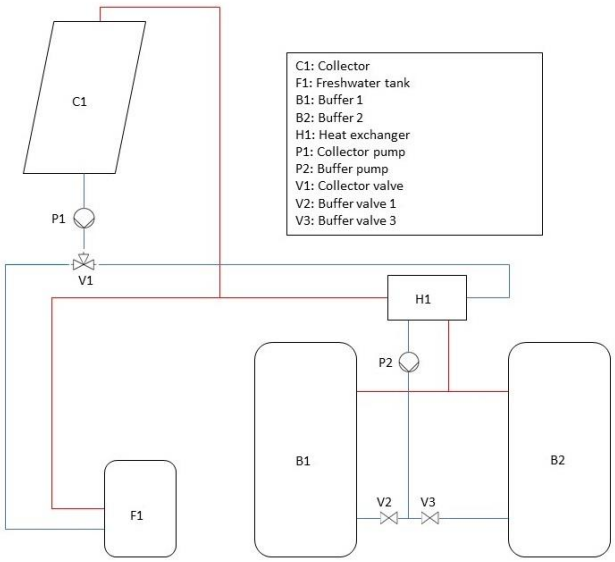

Figure 1: Integration of solar system

The future control unit must decide between four states: heating fresh water, heating buffer one, heating buffer two and no heating. Therefore, seven temperatures values are required: buffer 1 up, buffer 1 down, buffer 2 up, buffer 2 down, fresh water tank up, fresh water tank low and collector. Gathering this data is done by two different types of temperature sensors. The collector temperature is measured by a PT 1500, that was delivered by the solar systems vendor. This PTC (Positive Temperature Coefficient) is basically a resistor that changes its resistance depending on temperature. All the other values are measured by DS18B20 one-wire devices, that are cheap, precise and fast enough. The measuring range of these devices is between -55°C and +125°C.

Depending on the four states the different actuators (pumps and valves) must be switched on or off. This is done by a simple relays board, which matches the electrical requirements in terms of voltage and maximum current.

Connecting these components to the Arduino Uno board is not very complicated. The PTC-sensor is connected to GND, +5V and any analog pin. To measure its current resistance using the voltage divider formula another constant resistor is necessary. The optimal value for that additional resistor is about 2.5kOhm. This was calculated in an excel sheet by maximizing difference between the voltage over the PTC at 0°C and 100°C by using different resistors. The maximum difference is about 0.9V, that leads to 0.88mV per step on the Arduinos internal ADC (Analog Digital Converter). The one-wire devices need a +3.3V power supply. Two of them are connected to one one-wire bus. Each bus is connected to on digital IO pin. Each bus also needs a so-called pull-up resistor between data line and +3.3V. The relay board needs a +5V power supply and each relay is connected to a digital IO pin.

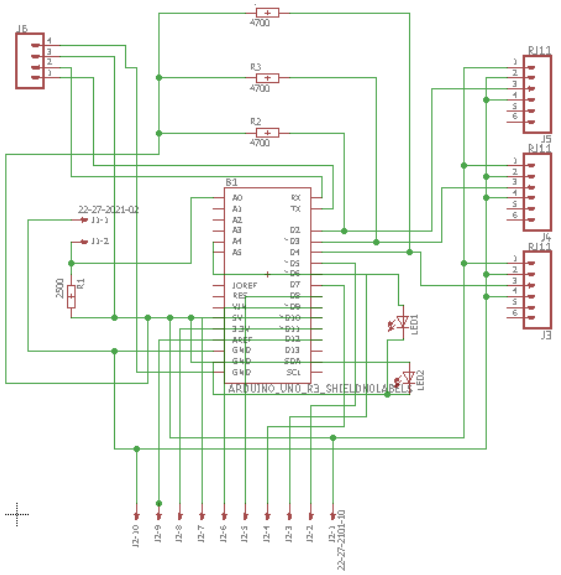

Figure 2 schematic Arduino shield

The most suitable way to manage the mentioned connections was to develop an own Arduino shield. The drawings and layouts therefore were done with a tool called “Eagle”[4]. After the layouts were finished the shield was manufactured and assembled in the appropriate facilities of the HTWG-Konstanz. Figure 2 shows the exact wiring on the new Arduino shield.

The software running on the Arduino board is a simple *.ino sketch. Getting the temperatures from the one-wire devices is easily done by existing one-wire libraries. For the PT1500 the value on the analog pin is direct proportional to the voltage over the resistor. That voltage can now be used to calculate the actual resistance of the PTC. With the given datasheet we can assume, that the temperature and resistance in the concerning range has linear correlation and can be described by the following polynomial formula: R =18.74 * X + 1630 where X is the current temperature and R the current resistance on the collector. After this calculation all necessary information is ready to use. According to the gathered information a function in the Arduino sketch is called and decides by checking multiple if-clauses, which of the four mentioned states will be activated. The activation of a certain state is done by an additional function, that sets the according digital IO pins to LOW or HIGH. LOW means that the according relay is triggered and HIGH the opposite. The final step in this code is to provide the temperature values to an interface for the monitoring system. In order to avoid using an additional ethernet shield for the Arduino the decision was made to use serial port. To guarantee a certain level of reliability and safety the whole loop must run through within a certain timespan. Otherwise a watchdog timer would immediately restart the program.

V. MONITORING SYSTEM

As shown in Figure 3 the monitoring system with graphical user interface consists of several parts.

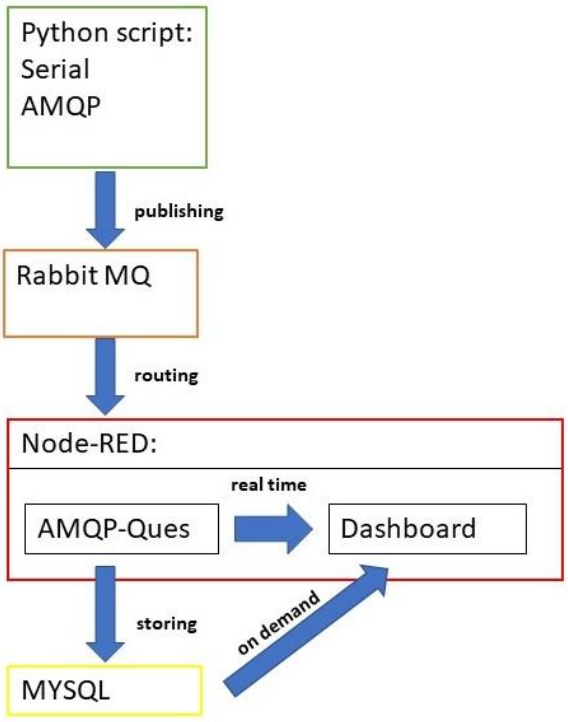

Figure 3 Architecture of the monitoring system

The python script connects to the raspberry’s serial port. Depending on which serial port is used, the connection between the two device is either done by an usb-cable or two extra digital pins. The raspberry communicates on a logic level of +3.3V and the Arduino uses +5V. In case of using the digital pins, an extra logic level shifter is required to set up the communication. After the communication is set up the python script parses incoming messages. For each temperature value and the current state of the control unit, a so-called routing key is defined. That routing key is part of the AMPQ (Advanced Message Queuing Protocol), that is used for transferring data in real time[5]. AMPQ offers many other routing options for example by a certain topic, which is similar to the MQTT (Message Queuing Telemetry Transfer). After defining the routing key, the sensor data is published to the AMQP-Broker, which is also running on the raspberry, but it could also run on a remoter server. The broker manages incoming messages by routing them to the different subscribers using the routing key. There are several implementations of AMQP-Brokers available on the market e.g. RabbitMQ or SwiftMQ. Since RabbitMQ is free to use, easy to install on the raspberry and offering a web dashboard for setup the decision was made to use RabbitMQ.

The next challenge towards achieving the goals was to visualize sensor data. Therefore, it would be nice not only display real time information rather to have a kind of archive where the user is able to have look on data incurred in the past. This case requires an additional database, that holds sensor data with the according timestamp.

Node-RED is a programming tool for wiring together hardware devices, APIs and online services. It provides a browser-based editor that makes it easy to wire together flows using a wide range of nodes in a palette that can be deployed to

its runtime. The runtime is built on Node.js taking advantage of its event-driven, non-blocking model. The flows created in Node-RED are stored using JSON which can be easily imported and exported. Additionally, JavaScript functions can be created within the editor using a rich text editor. For visualizing data there is a module available, that provides a set of nodes to create a live data dashboard, that can be accessed over the network with any internet-browser[6].

The Node-RED flow created for this project connects to the AMQP-Broker as subscriber, by using so-called AMQP-Ques, for the provided data. This data is then reformatted by several custom JavaScript functions and other nodes to be compatible with the mentioned dashboard nodes and the database. Live senor data are visualized in two variants. One variant is a simple gauge, the second variant is a graph showing data from the last few hours. Putting more data into that graph like multiple days would cause the dashboard to be very slow, that's why an external database is needed where the data can be loaded on demand. As database the decision was made to use a simple MYSQL implementation, since there are no real time requirements needed and the tabularly data format is enough for storing sensor data with timestamp. The database is also running on the raspberry but like the RabbitMQ broker it is possible to run it on a remote sever.

The last step is to make the dashboard accessible over the internet. Therefore, two steps are required. The first step is to use a dynamic domain name service is necessary to make the raspberry accessible with a constant name e.g. "www.solarsystem.dnshome.de" instead of knowing the exact IP address. The second step is to forward the standard http port 80 or 8080 to the raspberry's port where the Node-RED dashboard is running. To make sure, that only entitled users are able to access the dashboard the Node-RED installation needs to be reconfigured. After these steps the used internet-browser asks for username and password every time connecting on the dashboard.

VI. RESULTS

The web interface for monitoring can be accessed from the internet. Sensor data and the current system status (heating fresh water, heating buffer one, …) are updated in real time every few seconds. The archive function is also doing very well, the data acquisition from the database is fast but depends on the chosen timespan. The system is now tested for several weeks without any problems. Figure 4 shows two snapshots of the actual dashboard.

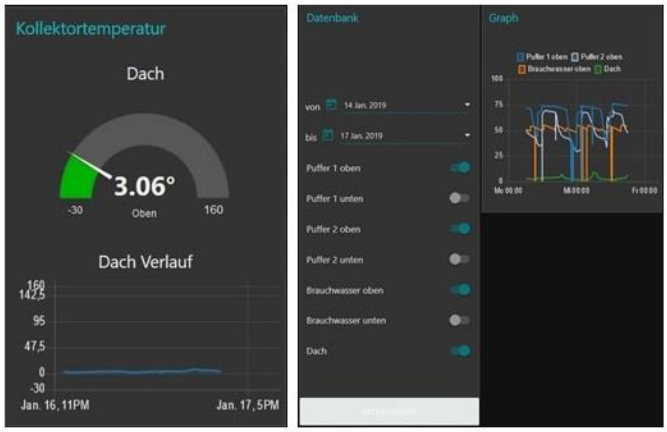

Figure 4 monitoring interface snapshots

The left snapshot shows the actual collector temperature, these days the collector is covered by snow so there is no chance to see the system in action. The right snapshot shows the archive view. With the date pickers the user can choose a certain timespan and with the sliders a selection of the seven sensor dates shown in the graph on the right. As seen in the rights snapshot's graph, there are some outliers but the only occur with the one-wire devices.

VII. CONCLUSION AND FUTURE WORK

These results show that novel tools and frameworks like Node-RED, Arduino, embedded Linux are suited for the fast development a control unit and monitoring prototype for solar thermal systems. Controls over internet and real time visualization of the sensor data recorded are offered through the help of a platform-independent web interface.

Future effort will be invested in adjusting the regulation parameters for the control unit during the summer months to improve efficiency of the solar system. Further there will be improvements on the monitoring interface site. Outliers will be removed. It is also planned to add some additional control features.

REFERENCES

[1] https://www.umweltbundesamt.de/daten/energie/stromerzeugung-erneuerbar-konventionell

[2] https://www.studentenergy.org/topics/solar-thermal

[3] https://www.energie-experten.org/heizung/solarthermie/solarthermieanlage/high-flow-anlagen-und-low-flow-anlagen.html

[4] https://www.autodesk.com/products/eagle/blog/arduino-shield-buying-designing/

[5] https://www.amqp.org/

[6] O'Leary, Nick (August 14, 2017). "node-red public". *nodered.org*. npm. p. 1. Retrieved Aug 20, 2018.