Storing and visualizing data using the Raspberry Pi

Felix Ragnarsson
Nurhusen Saleh
Erik Sundberg
Adam Woods
Abstract

Storing and visualizing data using the Raspberry Pi

Felix Ragnarsson
Nurhusen Saleh
Erik Sundberg
Adam Woods

This project concerns the development of a solution for visualizing and storing different types of data using a single board computer. This replaces a system that currently uses Windows-based laptops. To do this, a suitable single board computer has been selected, with price and performance in mind. The one selected is the Raspberry Pi, due to it having good performance for its cost and the external stakeholder desiring its use. For database functionality a MySQL system is implemented because of its network and concurrency capabilities. In terms of results, the external party’s performance requirements are that data from sixty different sensors must be read and saved once per second. These have been met. For visualization, various frameworks and related tools are implemented, and the user interacts with the system through a touch interface.
Sammanfattning

# Contents

1 Introduction .......................................................... 1  
2 Background .............................................................. 1  
   2.1 Database management .................................................. 2  
   2.2 Data visualization ....................................................... 2  
   2.3 Single board computers ............................................... 2  
   2.4 Operating system ...................................................... 3  
   2.5 Measuring unit .......................................................... 3  
3 Purpose, aims, and motivation ....................................... 3  
   3.1 Project aim ............................................................. 4  
   3.2 Delimitations ........................................................... 4  
4 Related work ............................................................... 5  
5 Method ............................................................. 6  
   5.1 Choice of single board computer ...................................... 6  
   5.2 Programming language ................................................ 6  
   5.3 Database ............................................................... 7  
6 System structure ........................................................... 8  
   6.1 Measuring nodes ........................................................ 8  
   6.2 Database structure .................................................... 9  
7 Requirements and evaluation methods ............................... 11  
   7.1 Performance ............................................................ 11
7.2 Database disruption .............................................. 11
7.3 Comparison between the old and new system .................. 12
7.4 Graphical user interface ......................................... 12

8 System description .............................................. 13
  8.1 Starting a session ............................................ 13
     8.1.1 Starting a session, front end .......................... 13
     8.1.2 Session back end ........................................ 14
  8.2 Display current session ...................................... 15
     8.2.1 Current session front end .............................. 15
     8.2.2 Current session back end .............................. 16
  8.3 End current session .......................................... 16
  8.4 Show old sessions .......................................... 16
     8.4.1 Old sessions front end ................................. 16
     8.4.2 Old sessions back end ................................. 16

9 Risk analysis and error handling .................................. 17
  9.1 Failure to connect to the main database ...................... 17
  9.2 Measuring unit failure ....................................... 19

10 Evaluation results ............................................. 19
  10.1 Evaluation of system performance .......................... 19
  10.2 Evaluation of database disruption .......................... 20
  10.3 Evaluation of old versus new system ....................... 20
  10.4 Evaluation of the graphical user interface .................. 21
     10.4.1 User 1 ................................................. 21
11 Results and discussion

11.1 Results

11.2 Discussion

12 Conclusion

13 Future work

13.1 Support for an additional display

13.2 Additional system feedback

13.3 Export database to .csv format
1 Introduction

If someone is running a Windows laptop for long-term testing and wishing they could have something small-scale and cheaper, this problem solves that. This is done by eliminating the laptops and replacing them with single board computers, running Linux.

The Raspberry Pi foundation is responsible for the single board computer series Raspberry Pi, which are high performance / low price computers. These small computers are powerful enough to replace larger, more powerful computers in certain tasks. In this project we have replaced laptops that are used to store and visualize data that come from sensors in long-term tests. This will be done in Raspbian, the officially supported operating system for Raspberry Pi hardware [12].

For data storage, two databases using MySQL [7] is implemented. One of these will be used for local storage that will save local measurements, and another database that will be used to store all measurements from multiple local databases. Visualizing the measurements is done using a graphical user interface (GUI), that will be able to display the measurements in e.g. a graph over time. The user will also be able to change which measurements that are shown by changing time interval, sensor and type of graph. The Qt framework will be used to design the GUI [2].

2 Background

APR Technologies (henceforth called APRtec) [5], a company based in Enköping, is the external party for this project. The company develops embedded systems solutions for the space industry. Their primary product areas are fire alerts, heat dissipation, sensors and surface materials.

The main areas that this project will incorporate are database management, data visualization, and single board computers, namely the Raspberry Pi single board computer. A basic understanding of each of these areas is useful when reading this report and therefore a brief introduction will be given to each of them. Furthermore, the measuring units used to collect data will be explained, as well as our choice of operating system for the single board computers.
2 Background

2.1 Database management

A database is, simply put, a structured collection of data. The data in a database is useful in some way, represents some real-world entity and can be anything from phone-numbers or names to images or books [16]. In this sense, a library, images on a hard drive or even a collection of physical movies can be seen as databases, but in the field of computing this refers to data stored on a memory unit.

The area of database management involves the creation, deletion, maintenance etc. of databases using a database management system. According to Ramez Elmasri, a database management system is defined as a general-purpose software system that facilitates the processes of defining, constructing, manipulating, and sharing databases among various users and applications [16]. In other words, a database management system can be seen as an interface to an actual physical database, allowing users to interact with it in different ways. There are a number of various database management systems available for use such as MySQL [7], SQLite [13] and Oracle [8].

2.2 Data visualization

Data visualization concerns the area of taking data and presenting it in a concise and useful way, depending on requirements. The simplest example are bar charts to determine some effect, to more complex visualizations such as three-dimensional scatter plots. There are many tools and frameworks for this available for Python, such as Matplotlib, Chaco or PyQtgraph. There are possibly performance issues to consider when selecting one or more of these. The basic requirement for visualization is to integrate it in a touch screen GUI.

2.3 Single board computers

Single board computers are small computers contained on a single circuit. They contain one or more CPUs along with memory and programmable input/output peripherals. Single board computers are used in embedded applications, in contrast to the micro-processors used in personal computers, ranging from basics such as equalizers to more advanced uses like cars or satellites. Their main advantage is cost-effectiveness and ability to perform many different tasks depending on user requirements.

The Raspberry Pi Foundation is a charity that was founded in 2009. The foundation has developed a series of single-board computers to promote computer science in schools.
These single-board computers have a high performance/cost ratio, are as big as a credit card, and are provided by the foundation [10, 14]. Like other single board computers the Pi can be used in many embedded systems areas, such as scientific experiments [18], home automation [17] and data analysis [21].

2.4 Operating system

Linux is a Unix-like operating system that contains free software. The group has chosen the Linux distribution called Raspbian for the project. This is the officially supported operating system for the Raspberry Pi [12]. Raspbian is a modified version of Debian, which is an operating system that uses the Linux kernel [11, 1]. The main reasons for using Raspbian is threefold: it comes with most or all of the software packages needed, built-in support for the Raspberry touch screen, and in the group’s opinion a larger chance of long-term support.

2.5 Measuring unit

In this project, the measuring unit will be considered a “box” that is a collection of sensors that sends its collected data via USB. In other words, we are not concerned about the internal functionality of the unit. The data sent by the measuring unit will be handled by our software, which is run on the Raspberry Pi’s. In total, sixty measuring channels are available. Each channel is connected to a sensor, that has the ability to measure real world physical phenomena.

3 Purpose, aims, and motivation

APRtec currently uses Windows laptops for their work. There are ready-made solutions available for these systems, but APRtec is unsatisfied with these due to stability concerns and general clunkiness of Windows. This is due to there being so much pre-installed software that comes with each installation. Every manufacturer likes to provide their own software for what they consider added functionality. Even if removing all of it is possible, there exists the possibility that one missed something. This could then interfere with system operation over time. The company has given examples of some program installing an update without asking after several weeks and thus interrupting the work the unit was doing. A single board computer would be more appropriate to use in terms of the above, physical size and with Linux being more stable [23]. Linux also does
3. Purpose, aims, and motivation

not include bloatware to the same degree (depending on distribution), and automatic software upgrades must be enabled manually by the user.

The overall question that this project aims to answer is how the Raspberry Pi single board computer can be used to store and visualize different types of measurement data. The project is important because it would result in a general purpose solution to the problem of storing and visualizing data. Many different solutions exist for the purpose of visualizing and storing data using conventional computers, but there is a lack of a general purpose solution using a cheap, small, portable system like the Raspberry Pi. Reaching the goal of this project would therefore allow individuals with a small budget to visualize and store data, something that is useful in many different fields of work.

3.1 Project aim

A software solution will be developed for the Raspberry Pi which will make visualization cheap and accessible compared to using traditional computers. An important part of the project is to build a solid code foundation that is properly standardized, commented and documented for future expansions. The expected effect of this is that companies and individuals will be able to use the solution when doing work involving any data they would like to use. Specifically, the project consists of taking measurement data from a measuring unit over a USB connection. This must be interpreted and presented in an intuitive way for APRtec’s employees.

The system will have a touch interface using the accompanying touch display that can be purchased for the Raspberry Pi. The system must be able to store data either locally in storage, or in a remote SQL database, depending on what the user wants.

Stability is an important part of the project, where data cannot afford to be lost, and if there is a power outage this must be handled, where the ideal situation is that the system resumes operations where it was. This will also be the center of a possible cluster of single board computers. Problems to solve include creating an SQL database on a Raspberry Pi, transferring data between units, ensuring maximum polymorphism of data handling, a robust interface and redundancy protections such as lost data or power outages.

3.2 Delimitations

The most significant delimitation that has been made for the system is that the measurement data that the system will be used for has to be represented as float numbers, due
to time restrictions. Another delimitation is the number of measurement values that the system can handle each read cycle. The system is limited to 60 values per Raspberry Pi (excluding the main database) and the reason for this limitation is that the external party wanted a system that could handle this exact amount of values. Since it is unclear at this time whether the system could handle more values or not, the group chose to limit the system to this number instead of trying to expand the system capabilities.

Additionally, the system is limited in how often reading input and storing values can be done. APRtec wants the system to be able to read and store values at least once per second, and since the capabilities of the Raspberry Pi are limited, the decision was made to limit the system to the time constraint set by the company.

Another limitation concerns different Linux distributions, where the project only guarantees functionality for Raspbian. All Linux distributions are based on the same kernel, and the project should work for any variant, but this can not be guaranteed. Also, the system has been developed for the Raspberry Pi 3 Model B. No proper testing has been done on any other version 3 models, nor older Pi’s. Older Pi’s have lower specifications and thus performance may or may not be an issue.

4 Related work

A solution which tries to solve the problem is currently deployed by APRtec. The solution uses a Raspberry Pi for the gathering of data and laptops installed with the Windows 10 operating system to store and visualize it. The main difference between that solution and this project is that this project solves the problem using only the Raspberry Pi, thereby eliminating the required use of laptops with Windows. This project also allows the user to configure the system in some ways such as naming databases and channels, choosing the location for storing, and choosing different ways to visualize the data. This is done using a GUI, something that is not implemented for the current system used by APRtec.

A similar project exists where the Raspberry Pi is used as a temperature and pressure logger [20]. A Raspberry Pi measures temperature and pressure using a single sensor and sends the data to a third-party service which hosts and visualizes it. The main difference between this project and that project is that this project aims to develop a system which can be used to store and measure data from multiple sensors in one location, from multiple locations and from different types of sensors. This project also aims to offer a system that does not rely on any third-party service.

Other work related to this project is any project making use of the Raspberry Pi single
board computer. One example mentioned in *Raspberry Pi User Guide* is to use the Raspberry Pi as a web server [24], something that is done as a part of this project. The author of the book describes a common structure of web servers and describes the process of implementing the structure on the Raspberry Pi. This involves installing the operating system Linux, the web server Apache, the database management system MySQL and the scripting language PHP [24]. Something similar is done in this project, but differs in the sense that the web server Apache is not used and that the scripting language used is Python instead of PHP.

5 Method

This section describes the methods of choice for reaching the specifications of this project. In particular, our choice of single board computer, programming language, and why these interact well together. Furthermore, the reasoning behind these choices is provided.

5.1 Choice of single board computer

The single board computer used is called Raspberry Pi model 3 B [9]. APRtec desired the brand to be used since the company had used it previously. The group reviewed the specifications for that model and decided that the desired performance should be attainable. The specific model is used because it is currently the most powerful Raspberry Pi computer on the market [15]. Compared to other single-board computers, the Pi 3 model B has a high performance / cost ratio, and has good overall performance [19].

The official operating system of the Raspberry Pi, Raspbian, also includes native support for touch screens. Allowing visualization and interaction via touch screen is deemed an important part of the project. Native support is an advantage during development, since no custom solution needs to be created, increasing productivity.

5.2 Programming language

The programming language of choice for this project is Python. There are several reasons for this decision. Firstly, it was the wish of APRtec since they had used it previously. Secondly, our conclusion is that there are no urgent primary memory (RAM) limitations to be considered for this project. As the Pi has 1 gigabyte of memory, there
is no need for making the code as memory efficient as possible, in which case C or a similar language would be a better choice. Another aspect considered is the productivity level. Python syntax is easy to learn, enabling production of working code quickly. Also, the source code produced in Python is typically compact and easier to understand, which is beneficial for anyone wanting to become familiar with the project source code. Moreover, Python has cross-platform support, which will allow the code to be ported to many different systems.

A drawback of using Python is that the language is interpreted, rather than compiled. This means that Python requires more hardware resources in order to run programs, than compiled languages like C or Haskell.

5.3 Database

Designing the database was done by first creating an entity-relational (ER) model, to get a good overview of the data that needs to be kept in the database [16, p. 221].

The finished ER model is then translated to a relational model, which we use to improve and implement the database [16, p. 59]. The database was designed and developed with normalization order three (3NF) [16, p. 501], in accordance with standard design methods.

All the measurements will be saved to the local and main databases. The local database will be used to store a set of previous measurements that will be for the visualization. The local database is considered a temporary storage of measurements because of limitations to the amount of memory. The main database will be used to store all of the measurements. This is done to make sure that the user can access all the measurements that have been made during a long-term session.

MySQL is used for both the local and main database. It is used on the main database since it allows multiple users to use the database simultaneously, by using a locking mechanism that can be configured to either lock an entire table or a row in a table. This allows the main database to be used by multiple measuring units for saving measurements [6]. It will also be used locally because we do not want to develop a separate solution for storing data.
6 System structure

The overall structure of the system consists of a MySQL database and a cluster of Raspberry Pis’, as can be seen in Figure 1. Each Raspberry Pi in the system can be thought of as the main part of a node which is part of a network that collects measurement data from multiple locations, and stores it in the main database. In section 8, System description, the GUI is described in detail.

![Figure 1: Overall structure of the system](image)

6.1 Measuring nodes

Each node in the system consists of a Raspberry Pi, a local database and a touchscreen used to interact with the system GUI. The nodes receive data from a measuring unit through the Raspberry Pi USB-port. The data is then added to a database located on the Raspberry Pi storage memory. If the user has chosen it, the data is also added to the main database located remotely. The communication between a node and the main database is done through a MySQL connection. The structure of a node can be seen in Figure 2.
6 System structure

Figure 2: Image showing the structure of a system node. The parts marked red are preexisting hardware used in the project and the parts marked green are developed as part of the project.

6.2 Database structure

The main database consists of five tables. The relational schema of these can be seen in Figure 3. The table called “Pis” holds information about all the Raspberry Pis connected to the system, information that is used to identify a node that has added data to the database. The table called “channels” holds information about all the channels connected to the system, as well as with which Raspberry Pi the specific channels are associated. This is useful for identifying from which unique channel that each measurement originated.

Another table called “Sessions” holds information about sessions started by a user. The information held about a session is the name of the session, the time and date the session started, and the time and date the session ended. The information in the “Sessions” table is used in the measurement table so that measured data can be grouped as a session, making it possible for a user to search the database for measured data using a session as search parameter.

The last table, called “session_channels”, holds information about configurations that
the user has made for a channel before the start of a session. This information includes unit and tolerance of measured data, which is used when visualizing data.

As stated earlier, a local database is part of each node. Each of these databases have the same structure as the main database, with the exception that the “Pis” table and all keys referencing its attributes have been removed. The reason for this is that only one Raspberry Pi is storing measurements in the database, and therefore there is no need to store information of the Raspberry Pis connected to the database.

![Figure 3: The relational schema of the main database.](image-url)
7 Requirements and evaluation methods

The requirements for each part of the system will be evaluated using different methods. Each testing method is suited for testing if specific parts of the system are deemed to meet the requirements. In this section, these requirements and tests are described and motivated.

7.1 Performance

APRtec stated the following requirements related to the performance of the end product:

- The system needs to handle up to 60 parallel measuring channels simultaneously.
- Data gathered by the sensors must not be lost. The system must be able to receive new values every second, losing no information. Also, the data must not be corrupted by our program.

In order to gauge whether or not the system meets these requirements, a stress test has been performed. The stress test is intended to ensure the above mentioned functionality, testing the system under the heaviest load specified above.

The test setting will consist of a node collecting data from a USB once per second, storing this data in the local database of the node. The data read from the USB will be a predetermined sequence of values, representing measured data from each of the 60 channels. This is close to the real world setting, where we also read data from a USB which was written by the measuring unit. However, with predetermined values, we are able to determine whether the data stored by our system has been corrupted or not. Tests with multiple nodes writing to the main database have not been done due to time constraints.

7.2 Database disruption

The system should be able to recover from failure of the main database. Failure of the main database includes a power outage, network failure etc. System recovery means that each node in the system should continue storing data locally in the case of a disruption of the main database, and add the data to the main database as soon as it is functional again. Storing locally is done for as long as there is storage available, with the system
warning the user once this becomes limited i.e. if storage memory is nearly full. No data should be lost in the process.

The reason behind these requirements is that the external party asked for the possibility of implementing the main database using a third-party hosting service. This means that the external party would have no control over the stability of the main database. The requirements mean that if stability issues occur with the third-party, the system should respond by continuing operating as usual.

To evaluate how the system performs when a disruption of the main database occurs, tests have been conducted. These will be conducted by starting measurement sessions, letting nodes store data for a while and then disrupting the main database before making it functional again. The time will be between ten minutes and several hours. The evaluation metrics for the tests are how well the system functions when the main database is disrupted i.e. if it crashes or not, as well as if it manages to add the stored data to the main database when functionality is restored.

### 7.3 Comparison between the old and new system

The main requirements in order for APRtec to see this system as an improvement over the old one is for it to be more cost effective and stable than the one APRtec are currently using. Cost effectiveness is evaluated by comparing the cost of the single board computers we used to the cost of laptops.

Stability, on the other hand, must be tested over extended periods of time with the systems running in parallel. Therefore, testing this is too far beyond the time frame of this project, and thus cannot be tested by us.

### 7.4 Graphical user interface

The GUI needs to be intuitive enough for our external party. In this context, that means that the user should be able to start a new session and display the data associated with it, requiring no additional guidance but from the user interface.

This requirement will be evaluated by having APRtec staff who are normally responsible for the existing system perform a usability-test, more precisely a think-aloud test [22]. The test will be conducted by having a staff member perform the tasks of starting a session, ending the session, and visualizing the session after it has ended, all while communicating questions and thoughts vocally. This will help with assessing if the
8 System description

The system consists of several parts, which interact together in order to unify the front and back ends. For each functionality introduced for the front end (GUI) in this section, descriptions of the back end functionality will be presented.

8.1 Starting a session

In order for the system to receive, store and visualize new data, a task has to be started through the user interface, namely the “Start new session” task.

8.1.1 Starting a session, front end

In the main menu of the user interface, as shown in Figure 4, the topmost option is to start a new session. This choice will result in the screen shown in Figure 5 below.

![Figure 4: The main menu of the GUI. The options are: Start new measurement, show measurements, help, quit.](image-url)
A session can be started locally or remotely. Then the user is prompted to choose what channels to include in the upcoming session. Furthermore, the user may choose a name and tolerance for each of the specific channels. Tolerance is a mathematical precision expression of the measurements. Once all the desired user input has been acquired, a graph associated to a single channel in the current session will be displayed in real time. In the drop down box seen in Figure 6, the user has the ability to choose what channel is to be displayed.

### 8.1.2 Session back end

Until the front end has received any kind of input from the user in Figure 5, the back end is idle, i.e. waiting for the GUI to wake it up. Once the GUI does so, the input will be written to the config file of the system. The back end now reads this information, and starts receiving measurements once per second from the specified channels. This data is stored in the local database, and then displayed to the GUI. In the GUI, the user can choose to also store the data in a remote database.
8 System description

Figure 6: The interface displays a graph window for visualizing the data being collected. The x-axis is time, the y-axis, in this case, voltage.

8.2 Display current session

The data being collected by a running session does not have to be constantly visualized. The system supports continuous interaction while data is being collected.

8.2.1 Current session front end

The user interface allows the user to navigate the rest of the GUI while a session is active. Should the user wish to see the graph associated with the session currently running, it can always be shown by clicking the second topmost option in the main menu, “Display current session”. This will result in the data being visualized in real time, identical to the graph shown in Figure 6.
8.2.2 Current session back end

The data associated with the current session will not be kept as a local variable in the program should the graph window be closed. Instead, as soon as the data of the current session needs to be displayed again, this data will be reacquired from the database where it is being stored.

8.3 End current session

Choosing this option will stop all the measurements from all the channels currently active in the session. This allows the starting of a new session. If there is any data that has not yet been sent to the main database for storage, the back end will send it.

8.4 Show old sessions

Sessions that have been started and then terminated may be shown by the user interface.

8.4.1 Old sessions front end

All sessions that has been stored, either locally or remotely, can be chosen in the drop down menu shown in Figure 7. The sessions are displayed in reversed chronological order from top to bottom.

In a similar manner to the graph presented in Figure 6, section 8.1.1, graphs corresponding to each specific channel will be presented on the display, once a session has been chosen. However, the graphs will not be displaying new values continuously, but only values stored from the fixed time interval that the session was active.

8.4.2 Old sessions back end

The old session could be either remote or local. In the case of local, the back end will simply acquire the data in the local database, using the start and end date specified by the user as a limit for how much data to get. Should the old session be stored in the main database, the same procedure is performed, with the difference being that the back end connects to the main database in order to acquire the data.
9 Risk analysis and error handling

When running the system, certain hardware and software errors may potentially occur. This section will describe the risks and how they are handled within the system.

9.1 Failure to connect to the main database

A failure of this type is either an internal problem at the hosting service of the database, or a network failure. These are problems that cannot be independently solved by our system. Failure to communicate with the main database would result in the following loss of functionality:

- No data can be sent to the main database.
- Old sessions can not be accessed and displayed.
- Remote sessions cannot be started.

Figure 7: The interface prompting the user to select a current, or a previously terminated, session.
Sending the data to the main database is handled by the system by saving a time stamp for the last time data was sent and received. Thus, once communication is established again, the data will be sent and no data loss occurs. Additionally, the user interface will display a red status icon and an error message once disruption of the main database occurs. This is intended in order for the user to be able to recognize the issue. Figure 8 below illustrates this functionality.

Figure 8: The warning signal that is displayed when the main database is unavailable. The warning text states that the connection to the database has been lost, and that the program will try to reconnect while the current session is being stored locally. It also says that the red warning button will disappear if the connection is successfully reestablished.

As for the loss of functionality regarding the lack of ability in starting remote sessions and accessing old sessions, there is no way for the system to handle this. A way to solve this issue could be to have another database acting backup for the entire main database. However, this would double the cost of having a main database, for the sole purpose of being able to use this functionality in the case of temporary downtime. Therefore, we deem this solution ineligible.
9.2 Measuring unit failure

The measuring unit could write the string containing the measurements in a faulty way. If it is faulty, the string cannot be parsed or casted into a float, which are necessities in order for the system to handle the values. An error like this is handled by the system by first throwing an exception. Then, the measuring unit will be signaled by the software to reset its sensors.

An error log will document what measuring unit caused the error, as well as the time it occurred. This information will be saved every cycle that the system cannot retrieve any new measurement values. For as long as no values are received, the status icon in the user interface will be red. This is similar to the error illustrated in Figure 8.

This allows any user to recognize whether the system is currently functioning as intended or not. A hard reset of the measuring unit might be required in order to fix the problem, and this feature allows recognition of that.

10 Evaluation results

Evaluation of the system was conducted using several testing methods. In this section, evaluation results for each distinct part of the system are discussed.

10.1 Evaluation of system performance

System performance was tested as is described in section 7.1. The duration of the test was set to 10 hours.

Regarding the requirement of no data loss, the test resulted in the system storing approximately 2 million values, which was the desired outcome. The integrity of the stored data was then confirmed by running a script that compared the values in the database, to those in the predetermined sequence. As all the values were intact, and the correct amount of values were stored, we conclude that the system could receive and handle 60 values per second. As these were the specified requirements for this part of the system, we deem these requirements met.
10.2 Evaluation of database disruption

When evaluating how well the system could handle a disruption of the main database, the evaluation method described in section 7.2 was used. In the first test, a session using sixty channels and a storing frequency of 1 Hz was started. After approximately one minute, the connection between the main database and the node used in the test was disrupted by disabling the WiFi used to maintain the connection. The WiFi was disabled for approximately ten minutes before being enabled again, and after another minute the test was terminated by ending the session.

In the second test, an identical session was started. After approximately one minute the connection to the main database was disrupted in a similar way as in the first test. The disruption was maintained for approximately one hour before a connection was established again, and after one minute the test was terminated by ending the session.

In section 7.2 it is stated that one of the system requirements related to a disruption of the main database is that the system should continue storing data locally when a disruption occurs. When looking at the entries made in the local database during both test sessions, it becomes clear that the requirement has been met. The entries show that one value has been added to the database each second during the entire duration of both of the tests i.e. the system has remained functional and added values even though a disruption occurred.

Another system requirement mentioned in section 7.2 is that when the main database becomes functional after a disruption, the data stored locally during that time should be added to the main database. After the first test had been conducted, the local database was compared to the main database. The result was that both of the databases had 35,460 entries and that the entries in both databases were identical. The fact that the entries were identical were confirmed by running the same SQL query on both databases and comparing the results with a simple script. The result of the second test was similar to that of the first test with the exception that 154,860 entries were made to both databases.

10.3 Evaluation of old versus new system

In the context of comparing the old system with the new one, the only requirement that could be evaluated within the scope of this project is cost of hardware. Specifically, the minimum hardware required in order to maintain a functional system, i.e. reaching all the requirements specified in section 7.

The Raspberry Pi 3 model b can, at the time of this report, be acquired for 36.55 USD [3]. Furthermore, a minimum of one touch screen must be included, and can
be bought for 69.95 USD [4]. Thus, the total cost of hardware for a minimal setup of our system is 106.5 USD.

This is weighed against the cost of a laptop using the Windows operating system.

10.4 Evaluation of the graphical user interface

The evaluation of the graphical user interface was conducted by having two of the end users perform a think-aloud test where the tasks given were to start a session, end the session and visualize the session after it had been ended. The results of the two think-aloud tests are presented individually below.

10.4.1 User 1

When given the task to start a new session, the user managed to perform the task without any major problems. However, one of the problems that the user faced was to click the checkboxes in the channel-selection table shown in 5. The user tried to click a checkbox and when it did not show any response, he voiced concern and wondered where to click to fill the checkbox.

After finishing the task to start a new session, the user was given the task to end the session and visualize it. The user had no problems ending the session or locating the button on the main menu to visualize it, but after choosing to visualize data from the main database he again voiced concern. The program was fetching data from the main database for a few seconds, but since no indication of this was given, the user believed that the program was malfunctioning. When the data had been fetched and the GUI was updated, the user suggested adding some type of indicator that the program was fetching data.

10.4.2 User 2

The user had no problems whatsoever when starting a session, and he had nothing noteworthy to say when performing this task. He also had no problems with ending the session or visualizing it but he had some opinions on how the data was visualized. He expressed a will to plot multiple plots in one graph and also wanted functionality to export plotted data to a csv-file.
11 Results and discussion

11.1 Results

The project result consists of a Raspberry Pi system that can read data over a USB connection, store these both in a local and a main SQL database, as well as showing the data in graphs. APRtech is able to replace the Windows laptops they currently use with our solution with a minimum amount of work, where they only need to configure the operating system and a few variables in the program in order to start using it. Using Raspbian Linux, stability and lack of bloatware is guaranteed, with no spontaneous updates to software being made without user input.

Performance is in line with requirements, storing measurements once per second, and storing them remotely if necessary. This defaults to every ten seconds, but can be changed by the user. The reason for not storing remotely every second is due to bottlenecking over the network, making it impossible to keep up with the current system design. Touch screen functionality is implemented, enabling the use of the official Raspberry Pi touch screen as intended. The GUI resizes dynamically in order to function on both a large and a smaller touch screen.

All goals for stability have been met. The system can ignore faulty data, such as strings with header information being transmitted, and avoid storing these. Power outages are also handled, with recovery done automatically on both a local and remote level. If a node powers down for some reason, once it has regained power it will continue taking measurements automatically. Similarly, if the central database node loses power, the local nodes will continue saving data locally until a connection is re-established, and then the remote transmission of data will continue as normal after storing any queued data.

11.2 Discussion

Overall, the project goals have not been overly difficult to achieve. Small ideas and considerations that the group had at first have been either changed or not implemented, but most of what the group initially envisioned is in place in the finished system. Development has been a smooth process, with the usual problems of getting frameworks running alone or with other frameworks, learning various technical aspects of the Pi. Choosing Python as the language was a good decision, for the reasons considered from the beginning. All group members have become proficient with writing Python, and agree that the language itself has presented no barriers in developing the system.
Overall, any problems encountered during the development process were due to group inexperience in a real-world situation, which has meant that every member has learned a tremendous amount about software development. The group re-designed the database five times, mostly due to inexperience, having only taken one course in database design, but also due to customer feedback. This was also expected, since it is a vital part of the system and specifications can easily change depending on user requirements.

Initially, one feature the group intended to include was support for two screens. This would have the GUI controls on the touch screen, and graphs on the other. Unfortunately, two screens cannot be used at the same time, due to the operating system only having one frame buffer. Documentation for Raspberry Pi framebuffers is limited, only mentioning one example program that has implemented a second buffer. Searching the web makes it clear that Raspbian does not support this. There exists other software where the creators have constructed their own frame buffers. This is however beyond the project scope.

In terms of what could have been done better, the clearest example is SQL. The group had pre-conceived ideas about what to use without doing proper research into which variants were good for what situations. SQLite was considered for local storage, simply because the group has a notion of it being light-weight, implying it would be suitable for a single board computer. Had the group read more about SQLite it would have been clear that there is almost no reason to use SQLite over MySQL today, since most things are faster in MySQL. Also, some libraries were difficult to install for certain versions of the Python interpreter, where a lot of times was wasted trying to get these to work. Getting a MySQL connector working on a Mac was very difficult and frustrating, but was solved by using PyMySQL which worked for all systems. Later in the project, coding was concentrated to just one or two members of the team, while the rest worked on this report. This makes sense in terms of efficiency, but more principally all members need to be familiar with what makes the project work, and desire to have contributed to the project on a technical level.

12 Conclusion

All the important requirements for APRtec have been fulfilled. APRtec now has a system that can replace their Windows laptops for storing measurement data, using MySQL for local and centralized database storage. This uses a single board computer, the Raspberry Pi, with a Debian-based operating system. These units, which are ordered in nodes consisting of a single board computer with a local database and a touch screen, can be connected to any measuring unit that sends data through a USB connection. APRtec
will be able to expand the measurement systems to many nodes without incurring large
costs, due to the low cost of the single board computer compared to the laptops. The sys-
tem has redundancy protection, handling data errors, database failures and also extreme
problems like power outages. The system uses a GUI with support for touch screens.
By using Python, they can use the system on most popular platforms, and due to exten-
tensive documentation and communication between the team and APRtec, they have a
good idea of what and how they can extend the system according to their own desires.

13 Future work

While this project has resulted in a fully functional product, there are several parts of the
system that could benefit from further development. Some of the functionality described
in this section was intended to be included by the end of this project, but could not
be implemented due to time constraints. The same applies to the feedback from the
usability evaluation of the GUI.

13.1 Support for an additional display

At the starting phase of this project, the group intended to implement support for show-
ing the user interface on both a touch screen and another screen via HDMI simultane-
ously. Once research into this began, it was quickly discovered that this would be very
difficult due to system constraints. Thus, if official support for multiple screens, where
one is a touch screen, is released, adding support for this will be a trivial task. This
would be ideal, since APRtec could use the touch screen merely as a controller, and
show all the data on the main screen.

13.2 Additional system feedback

The system does, in its current version, warn the user in the case of database disruption
as described in section 9.1. However, should the user not be able to interact with the sys-
tem via the touch screen for an extended period of time due to e.g. vacation, additional
warnings might be in order.

A possible enhancement would be sending an automated email to the user, should the
system not be able to backup its data for an extended period of time. Similar emails
could be sent in the case of the local database running out of storage space. This would
provide additional system feedback, and allow the user to, for instance, expand storage space before the system runs out of it.

13.3 Export database to .csv format

Exporting tables from a database to .csv format was intended to be implemented from the start. This functionality would be useful for using the data in other programs i.e. Microsoft Excel.
References


References


