Arduino Data Acquisition System for Monitoring Quality of Life Parameters in a Room

Alexandru MATEI[1]

[1]*Computer Science and Electrical and Electronics Engineering Department, Faculty of Engineering, "Lucian Blaga" University of Sibiu, Romania*

Abstract

Data acquisition is the process of collecting information from various sources through sensors or instruments. After obtaining the data, it is usually converted from the analogue format to a digital format.

In this paper an application that gathers room-related information is presented. The purpose of such a system would be to be able to monitor certain parameters of interest in a living space. Afterwards this data can be used in various manners. One of the most important results of this data is being able to identify the habitability of a room. It can point out potential health hazards before they cause actual health problems. An additional objective is for the system to be on an accessible budget and to be easy to understand and implement.

The hardware system using Arduino is presented in detail, as well as the graphical user interface created in Visual Studio.

The system works as intended for the purposes stated above. It measures: temperature, humidity, light intensity, sound, and air quality. It works on its own if it is supplied with a 9-volt power source. The Graphical User Interface can be used to visualise and analyse the collected data. The system can still be improved by adding wireless communication such as Wi-Fi and allowing it to be integrated into Internet of Things systems. The power consumption could be improved if needed for higher sampling frequency demands.

Keywords: Arduino, data acquisition, quality of life, habitability

1 Introduction

1.1 Description of the project

This paper presents a data acquisition system using the Arduino platform. The intent is to create an inexpensive system that is relatively easy to implement and understand. This is where the Arduino system shines, as it provides an abstraction between the hardware, the microcontroller, and the programming environment. This abstraction is, however, entirely optional, the ATmega328 controller of the Arduino being able to be programmed in a more traditional manner as well, if desired. In addition, the supplemental hardware used should be easy to come by or replaced if needed. Another convenient aspect is the community built around this platform. Due to its popularity the Arduino development environments are well documented. This makes it even more accessible than other possible alternatives.

From a hardware point of view, we just have the Arduino Uno Rev3 board, a microSD card module, the card itself, a RTC (Real Time Clock) module, a display made of 7 segment displays, an infrared receiver paired with a remote, and various sensors. The sensors record the following parameters: temperature, humidity, light intensity, sound, air quality, and the position of a potentiometer. Thus, the system is used for monitoring various parameters of a living space.
In addition to the hardware system, a software GUI (Graphical User Interface) was implemented using Microsoft's Visual Studio Integrated Development Environment (IDE). The program was built for a Windows 64-bit operating system. The application shows the acquired data in real time both visually, as a graph, as well as numerically, in a table. The application also allows configuring the system, saving the recorded data in a file, and loading a previously saved file. Furthermore, the application also has a serial console where the user can see the raw message sent by the microcontroller.
The interface allows for the following operations:

- Choosing and enabling the parameters to be measured;
- Setting the sampling frequency;
- Selecting the operating mode of the system: online or offline;

Also, the programs allows the user to save the configuration parameters in the EEPROM memory of the controller. If they are not saved, on the next power on, the microcontroller will load the old data saved in the EEPROM.
Overall, the system has 2 operating modes: paired , connected through an USB-A male to USB B male cable, or offline supplied with 9 volts either from an electric socket or, by using an adapter, from a 9 volt battery.

1.2 Topic relevance

The purpose of this work is to reflect the importance of data acquisition system and open-source hardware projects. By each passing day IoT (Internet of Things) systems play a bigger and bigger role in our society. In addition, the importance of data is increasing by each passing day, making data acquisition systems even more relevant. Platforms such as Arduino that cater to new technologies and concepts such as IoT are proving to be more and more relevant in our technological ecosystem. Arduino are already offering boards with Wi-Fi, Lora, GSM, and other technologies which helps them get chosen for IoT applications.

1.3 Objectives

The principal objective of this work is to create a data acquisition system that can process different parameters given from sensors. The system should be able to keep track of the date and hour of the data entries. In addition, the system should be able to work both paired, connected to a computer running a GUI, as well as offline, completely independent from external systems. In both modes, the system should confer to its users the option to configure the system as well as provide feedback after each provided command. Therefore, there are 2 main objectives: creating the hardware system and creating the GUI for the user. For the hardware part it is important for certain parameters to be chosen that reflect the functionality of the

system. As for the software, it should allow some standard features such as: saving and loading data, visualising data, and being able to configure the hardware system.
note

2 Project overview

2.1 Project presentation

The project was done in 2 big steps: developing the physical acquisition system and realizing the graphical user interface.
For the purposes of creating the hardware system the following components were used:

- The Arduino Uno Rev3 board, based on the ATmega 328 microcontroller from Microchip;
- A USB A cable male to USB B connector;
- The following sensors:
  - DHT22 – sensor module for measuring temperature and humidity;
  - MQ135 – sensor module for measuring air quality;
  - KY-038 – sensor module for measuring sound;
  - A photoresistor for measuring light intensity;
  - A linear rotary potentiometer of 50k Ohms;
- An infrared receiver and an infrared remote;
- A display made from 5 7 segment display elements and a 74HC595 8-bit shift register;
- DS3231 – a Real Time Clock module;
- A microSD module and the card itself (a 32 GB variant from Kingston);
- A 9-volt adapter accompanied by a 9-volt battery.
- A microSD USB card reader used for transferring the files generated by the microcontroller, in the offline operating mode, to the computer;

Following the hardware step, the microcontroller was programmed using the Arduino IDE. The following functionalities were added:

- Executing the commands given either by the interface or the remote control;
- Providing feedback when receiving a command, either through the serial port or through the display, depending on the operating mode;
- Implementing the communication protocol used to talk with the computer interface;
- Creating a data packet that contains the sensor data bundled with the date and hour at which said data was recorded;
- Either sending the data packets generated to the user interface, through the protocol, or saving it locally, on the microSD card;

The graphical user interface was done in Microsoft's Visual Studio IDE and programmed in the C# programming language. It has the following features:

- Communication protocol implementation to communicate with the microcontroller;
- Being able to configure the hardware system;

- Being able to view the received messages through a custom implementation of a serial interface;
- Graphs and tables to visualize the data from the microcontroller, both in real time and from loaded data;
- Saving and loading data in CSV (Comma Separated Values) or XLSX files;

3 Implementation details

3.1 Hardware implementation

In the Fig. 1 the block diagram of the hardware system can be seen. Here it is shown how the Arduino Uno Rev3 board is connected to all the other hardware elements of the system. The 4 analogue sensors (light, air quality, position, and sound) are connected to the analogue inputs of the Arduino. The temperature and humidity sensor are connected to a digital input, as the module is digital. The infrared receiver as well is connected to a digital input. The DS3231 RTC module is connected to the I²C interface. The microSD module is connected to SPI. Of particular importance is the 5-digit 7 segment elements display. Here we used a 74HC595 8-bit shift register to reduce the number of digital inputs needed on the Arduino to only 3 pins. Otherwise, the board would not have enough digital inputs for the display.

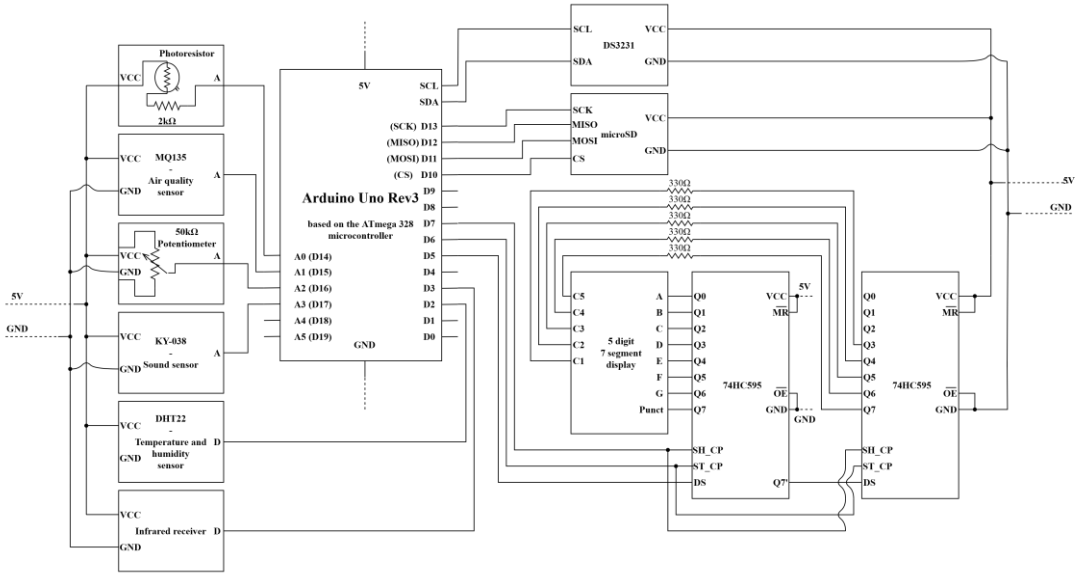

Figure 1. Block diagram of the hardware system

3.2 Communication protocol

3.2.1 System structure

This chapter has the role of providing an overview of how the system works. I chose presenting the system through the communication protocol used because this is what

ties the hardware system with the graphical interface. In Fig. 2 and Fig. 3 the overall structure of the system and data flow of can be observed. The system can operate in 2 different modes, online, or paired, and offline.
In the online mode, the system is connected directly to a computer through a USB cable. The Arduino transmits data to the computer through serial, the interface then displays the received data in graphs and tables.
In the offline mode, the hardware system is decoupled from the graphical interface. In this mode it must be supplied with 9 volts. Instead of actively transmitting data, while offline the system saves the data locally on a microSD card in a CSV file. Afterwards this file can be taken and loaded manually in the graphical interface for review.

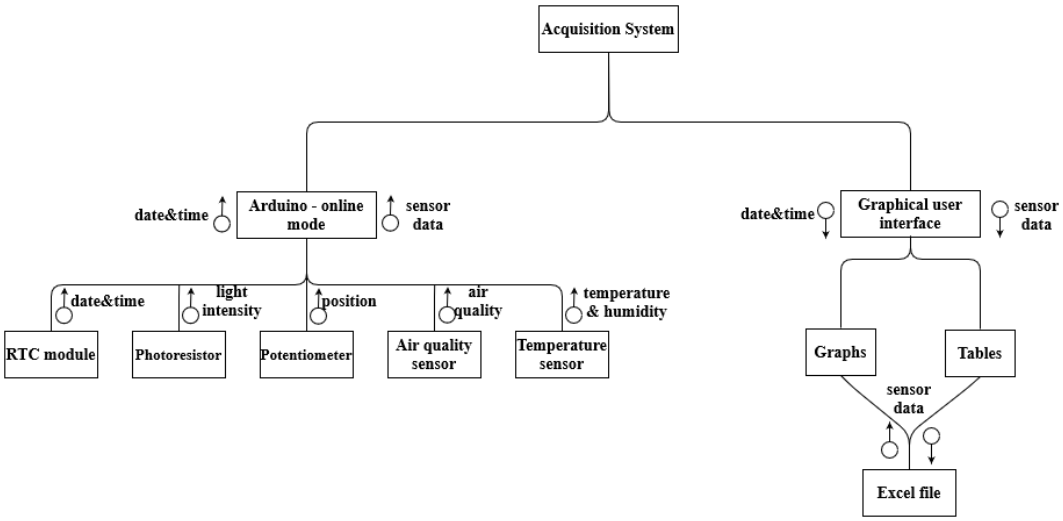

Figure 2. System structure – online mode

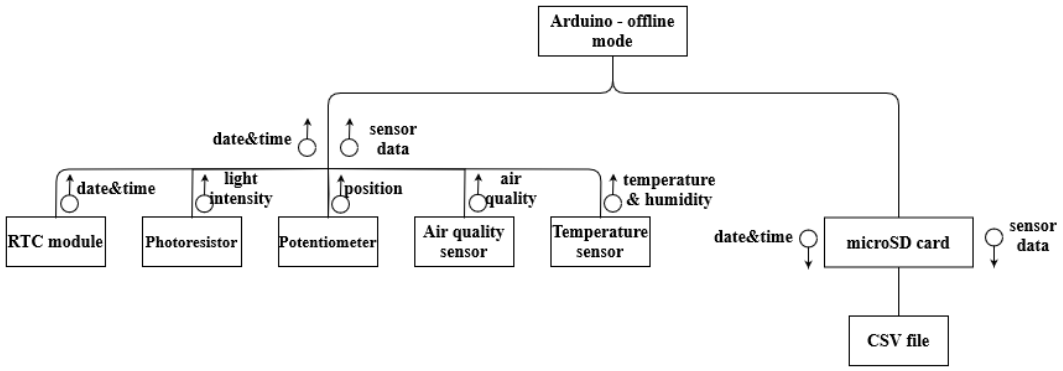

Figure 3. System structure - offline mode

3.2.2 Data transmission

The communication protocol used is built on top of the UART communication protocol. Over UART a secondary protocol was created for communication of the data acquisition system. It is represented through the structure of the data packets sent and the system configuration commands. The packets were designed to be compatible with the CSV file type. Therefore, each comma inside the packet indicates the end of a value and the beginning of a new one. The data packets have the following format:

YYYY-MM-DD,HH-MM-ss-mmm,T----,U----,S---,P---,A---,L---

As it can be observed, the packet has 8 different values. The first 2 are the date and time while the following 6 are values obtained from the sensors.

- YYYY-MM-DD represents the year, month, and day when the data packet was generated;
- HH-MM-ss-mmm represents the hour, minute, second and millisecond when the data packet was generated;
- T---- is the value of the temperature, which is represented with 1 decimal point precision;
- U---- is the value of the humidity, represented with the same precision;
- The following values indicate the sensor and the value generated by it in the interval 0 – 999:
    - S---- is the value of the sound;
    - P---- is the value of the position;
    - A---- is the value of the air quality;
    - L---- is the value of the light intensity;

The various sensor used have a different sampling frequency. The data acquisition system works at a sampling frequency of maximum 10 data packets per second. However, not all sensors used can keep up with this pace. According to the data sheet, the temperature and humidity sensor DHT22 has a sampling frequency of 0.5 Hz, which means it measure once every 2 seconds. The system was configured not to collect data from this sensor unless at least 2 seconds have passed from the last obtained value. If the overall system has a sampling frequency greater than 0.5 Hz and the DHT22 sensor has not generated a new value, in the data packet for the parameters of this sensor a null value will be transmitted. In Fig. 4 the serial console of the graphical application can be seen with data packages received in real time.

3.2.3 System configuration – graphical user interface

Depending on the operating mode of the system, it can be configured in one of two methods. In the online, paired, mode, the configuration can be done directly in the interface by using the buttons in the GUI. The application sends serial messages to the controller which then executes the configuration commands it receives. In the offline mode, the system can be configured by using the remote control.

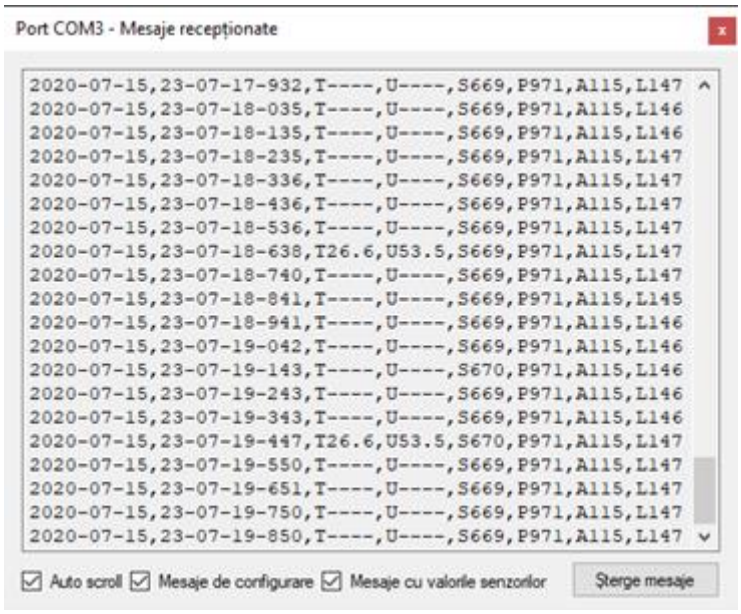

Figure 4. Serial console – viewing data packages

Here are the configuration commands that the controller has programmed into it:

- Config – asks the microcontroller for the configuration data saved within its EEPROM memory;
- Online – if possible, the controller switches to the online operating mode;
- Offline – the microcontroller switches to the offline operating mode;
- Start – begin data transmission;
- Stop – stop data transmission;
- CanTUSPAL – activating or deactivating certain channels. The letters 'TUSPAL' indicate the positions where, writing a 0 or a 1 enables or disables the corresponding sensor. For example, Can111111 is used to enable all the channels;
- A numerical value in the interval [0, 10]
    - This represents the sampling frequency in message per seconds. For example, to transmit a message once every minute we will give the command 0.01666;
- Salveaza – saves the configuration data in the EEPROM memory of the microcontroller;

Initially, when the microcontroller is first powered on, it is in the idle mode where it does not transmit data. Data transmission must be explicitly enabled.

3.2.4 System configuration – remote control

For the remote-control configuration of the system, the display comprised of 7 segment elements is important. It gives feedback to the user that the commands are received. In Fig. 5 are some of the messages the display can show. In order, they represent the following commands: Arduino online mode, Arduino offline mode, data transmission enabled, data transmission disabled, remote selected channel enabled, remote selected channel disabled, configuration saved, pre-set sampling frequency 01 enabled, and finally an error message.

Figure 5. 7 segment display command feedback messages

3.2.5 Communication protocol implementation

In Fig 6. and Fig. 7 the control flow and logic of the software programs can be seen. For the microcontroller software, the diagram is valid for both online and offline modes, the single difference being the different mode of transmitting the data packets. Otherwise, the behaviour is identical.

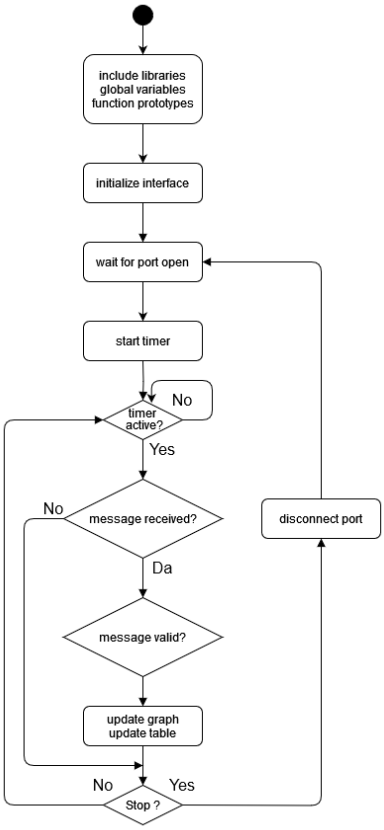

Figure 6. Activity diagram of the interface logic

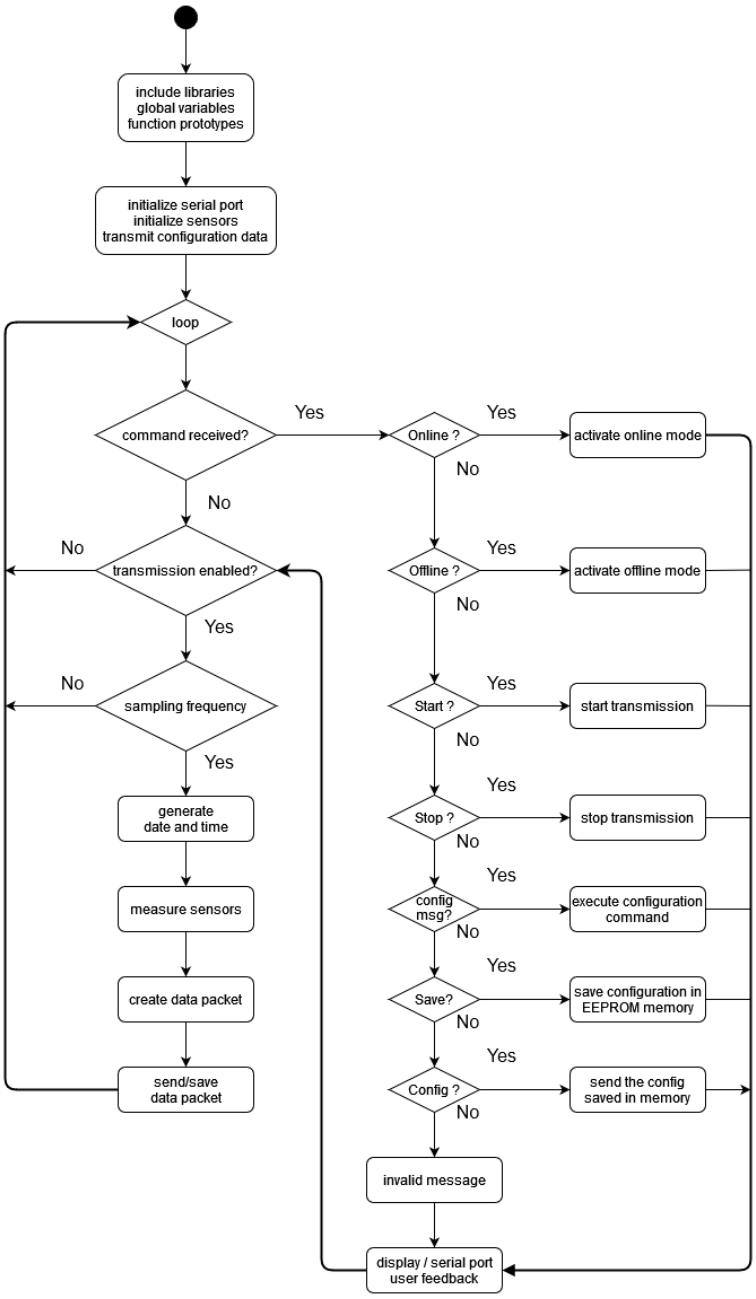

Figure 7. Activity diagram of the microcontroller software

4 Conclusions

The system works as intended and can be a suitable solution for tracking parameters of interest in a living environment. It can operate in two modes: online, or paired, and offline. The system gathers data from sensors measuring various parameters from a room: temperature, humidity, light intensity, sound, and air quality.

One disadvantage with the current implementation is the power usage. Configuring the system with a high sampling frequency consumes a lot of power, in this state the system can't be used for a long period of time. The system can be improved by optimizing power usage, increasing power supply, as well as adding additional

functionality such as Wi-Fi, Bluetooth, or other wireless communication protocol. With internet communication capabilities such a system can be used and integrated with other Internet of Things systems to provide additional benefits.
An advantage is the lower budget, as the resources of the Arduino Uno Rev3 were used to their fullest for this project, without being wasteful. The project could also be changed to use an ASIC (Application Specific Integrated Circuit), using just the ATmega 328, to reduce costs. But for the purposes of making an accessible, easy to use and understand system, that would hinder that goal.

References

[1] DUKA Adrian-Vasile, JOVREA Titus, *Sisteme cu microprocesoare. Microcontrolerul PIC18F4455*, Universitatea "Petru Maior" din Târgu Mureş, Târgu Mureş, 2010.
[2] GERMÁN-SALLÓ Zoltán, *Achiziţia şi prelucrarea datelor: Curs*, Universitatea "Petru Maior" din Târgu Mureş, Târgu Mureş, 2008.
[3] Jeremy Blum, *Exploring Arduino: Tools and Techniques for Engineering Wizardry 2nd Edition*, John Wiley & Sons, Inc., Indianapolis, ISBN 978-1-119-40537-5, 2019.
[4] Steven F. Barrett, Daniel J. Pack, *Microcontrollers Fundamentals for Engineers And Scientists (Synthesis Lectures on Digital Circuits and Systems)*, Morgan & Claypool, California, ISBN 978-1598290585, 2006.
[5] Steven F. Barrett, *Arduino Microcontroller Processing for Everyone!: Third Edition (Synthesis Lectures on Digital Circuits and Systems) 3rd Edition*, Morgan & Claypool, California, ISBN 978-1627052535, 2013.