# HARDWARE COMPOMENTS USED

## MICROCONTROLLERS. DEFINITION. STRUCTURE.

“ A microcontroller (or MCU for microcontroller unit) is a small computer on a single integrated circuit. In modern terminology, it is similar to, but less sophisticated than, a system on a chip or SoC; an SoC may include a microcontroller as one of its components. A microcontroller contains one or more CPUs (processor cores) along with memory and programmable input/output peripherals. Program memory in the form of ferroelectric RAM, NOR flash or OTP ROM is also often included on chip, as well as a small amount of RAM. Microcontrollers are designed for embedded applications, in contrast to the microprocessors used in personal computers or other general purpose applications consisting of various discrete chips.

Microcontrollers are used in automatically controlled products and devices, such as automobile engine control systems, implantable medical devices, remote controls, office machines, appliances, power tools, toys and other embedded systems. By reducing the size and cost compared to a design that uses a separate microprocessor, memory, and input/output devices, microcontrollers make it economical to digitally control even more devices and processes. Mixed signal microcontrollers are common, integrating analog components needed to control non-digital electronic systems.

Some microcontrollers may use four-bit words and operate at frequencies as low as 4 kHz, for low power consumption (single-digit milliwatts or microwatts). They will generally have the ability to retain functionality while waiting for an event such as a button press or other interrupt; power consumption while sleeping (CPU clock and most peripherals off) may be just nanowatts, making many of them well suited for long lasting battery applications. Other microcontrollers may serve performance-critical roles, where they may need to act more like a digital signal processor (DSP), with higher clock speeds and power consumption.”[ <https://en.wikipedia.org/wiki/Microcontroller>]

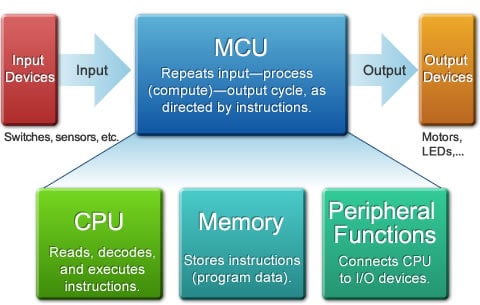


Figure Microcontroller structure

CPU: The "Thinker"

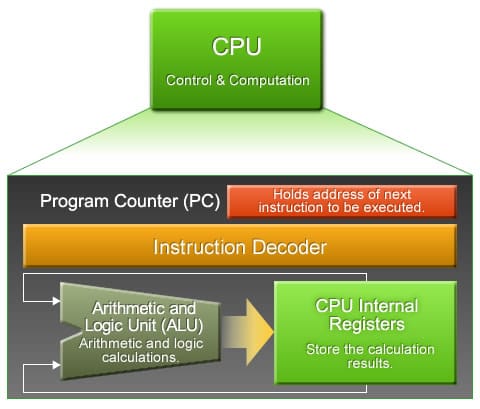


Figure Role of the CPU

* “ Program Counter (PC)

The program counter (PC) is an internal register that stores the memory address of the next instruction for the CPU to execute. By default, the PC value is automatically incremented each time an instruction executes. The PC starts at 0000, so the CPU starts program execution with the instruction at address 0000. As the instruction executes, the PC automatically advances to 0001. The CPU then executes the instruction at 0001, the PC advances again, and the process continues, moving sequentially through the program.

* Instruction Decoder

The decoder circuitry decodes each instruction read from the memory, and uses the results to drive the MCU’s arithmetic and operational circuitry. An actual decoder is a somewhat more complicated version of the decoder circuitry we studied in the session titled "Introduction to Digital Circuits-Part 2". It restores the encoded instructions to their original, unencoded form.

* Arithmetic and Logic Unit (ALU)

This circuitry carries out arithmetic and logical operations. Arithmetic operations include addition and multiplication; logic operations include AND, OR, and bit shifts. The ALU is controlled by the instruction decoder. In general, the ALU consists of a complex combination of circuits.

* CPU Internal Registers

These registers store transient information. General-purpose registers hold results of arithmetic and logical operations, whereas specialized registers store specific types of information—such as a flag register, which stores flag values (carry flag, etc.). When the ALU performs an operation, it does not operate directly on values in memory; instead, the data at the specified memory address is first copied into a general-purpose register, and the ALU uses the register content for the calculation.

Memory: The "Store"

The MCU’s memory is used to store program code and data. There are two main types of memory: ROM and RAM.

* ROM (Read-only memory)

This memory retains its content even while power is off. This memory is for reading only; it cannot be erased or overwritten. ROM is typically used to store the start-up program (executed immediately after power-on or reset), and to store constant values that may be freely accessed by running programs.

Many Renesas MCUs use flash memory in place of ROM. Like ROM, flash memory retains its content even while power is off. Unlike ROM, this content can be overwritten.

* RAM (Random-access memory)

This memory can be freely rewritten. Its disadvantage is that it loses its content when the power goes off. This memory is mainly used to store program variables.

Many single-chip MCUs (\*1) use static RAM (SRAM) for their internal RAM. SRAM offers two advantages it supports faster access, and it does not require periodic refreshment. The disadvantage is that the internal circuitry is complex, making it difficult to pack large quantities on the chip’s limited space. SRAM is not suitable for implementing large memory sizes.

The alternative to SRAM is DRAM (dynamic RAM). The simple structure of DRAM allows large quantities to be mounted in small spaces; typical DRAM sizes are much bigger than typical SRAM sizes. But it is difficult to form DRAM together with high-speed logic on a single wafer. For this reason, DRAM is generally not used within single-chip MCUs. Instead, it is typically connected to the chip and treated as peripheral circuitry.

(\*1) An MCU implemented on a single LSI (large scale integration) chip. The chip holds the CPU, some ROM, some RAM, oscillator circuitry, timer circuitry, serial interfacing, and other components. If the chip also includes the system’s main peripheral circuitry, it is called a "system LSI." “

[https://www.renesas.com/en-us/support/technical-resources/engineer-school/mcu-01-basic-structure-operation.html]

“ List of common microcontrollers

* Renesas Electronics
* Atmel
* NXP Semiconductors
* Intel
* PIC “

[https://en.wikipedia.org/wiki/List\_of\_common\_microcontrollers]

### ATmega2560 microcontroller

“The high-performance, low-power Microchip 8-bit AVR RISC-based microcontroller combines 256KB ISP flash memory, 8KB SRAM, 4KB EEPROM, 86 general purpose I/O lines, 32 general purpose working registers, real time counter, six flexible timer/counters with compare modes, PWM, 4 USARTs, byte oriented 2-wire serial interface, 16-channel 10-bit A/D converter, and a JTAG interface for on-chip debugging. The device achieves a throughput of 16 MIPS at 16 MHz and operates between 4.5-5.5 volts.

By executing powerful instructions in a single clock cycle, the device achieves a throughput approaching 1 MIPS per MHz, balancing power consumption and processing speed.

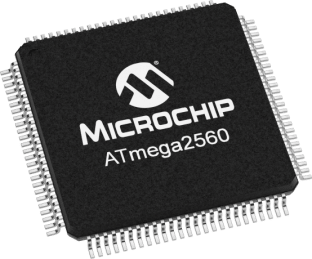


Figure ATmega2560 microcontroller

Parametrics

* Program Memory Type: Flash
* Program Memory Size (KB): 256
* CPU Speed (MIPS/DMIPS): 16
* SRAM Bytes: 8,192
* Data EEPROM/HEF (bytes): 4096
* Digital Communication Peripherals: 4-UART, 5-SPI, 1-I2C
* Capture/Compare/PWM Peripherals: 4 Input Capture, 4 CCP, 16PWM
* Timers: 2 x 8-bit, 4 x 16-bit
* Number of Comparator: 1
* Temperature Range (C): -40 to 85
* Operating Voltage Range (V): 1.8 to 5.5
* Pin Count: 100” [https://www.microchip.com/wwwproducts/en/ATmega2560]

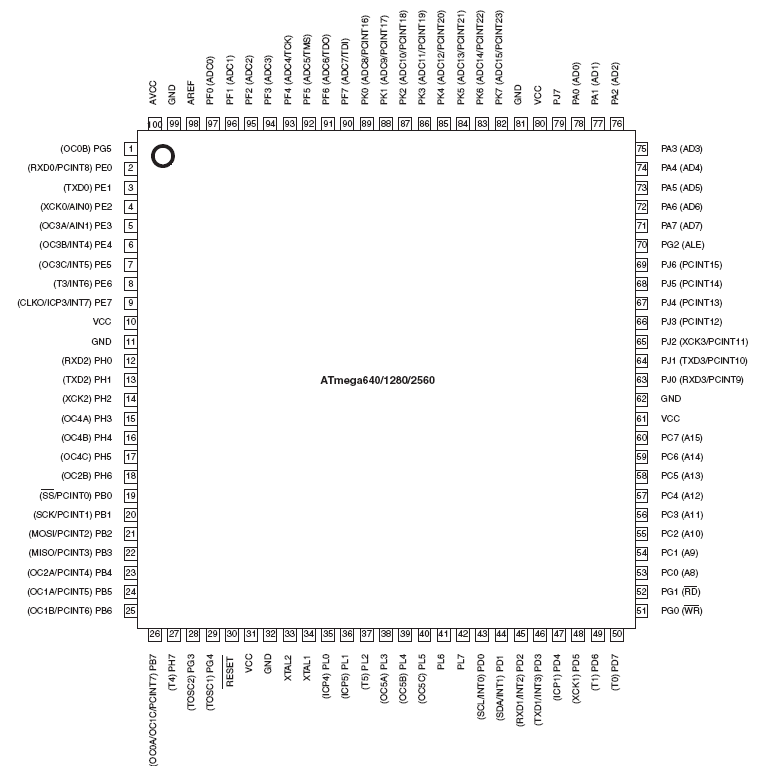


Figure ATmega2560 pin configuration

### Arduiono microcontrollers

“Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers - students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of accessible knowledge that can be of great help to novices and experts alike.

Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. As soon as it reached a wider community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IoT applications, wearable, 3D printing, and embedded environments. All Arduino boards are completely open-source, empowering users to build them independently and eventually adapt them to their particular needs. The software, too, is open-source, and it is growing through the contributions of users worldwide.



Figure Arduino Logo

Arduiono boards:

* Arduino UNO (Atmega328);
* Arduino Mega (Arduino Mega2560):
* Arduino Leonardo (Atmega32u4);
* Arduino Diecimila (Atmega168);
* Arduino Nano (Atmega328);
* Arduino Duemilavove (Atmega168); “ [https://www.arduino.cc/en/Guide/Introduction]

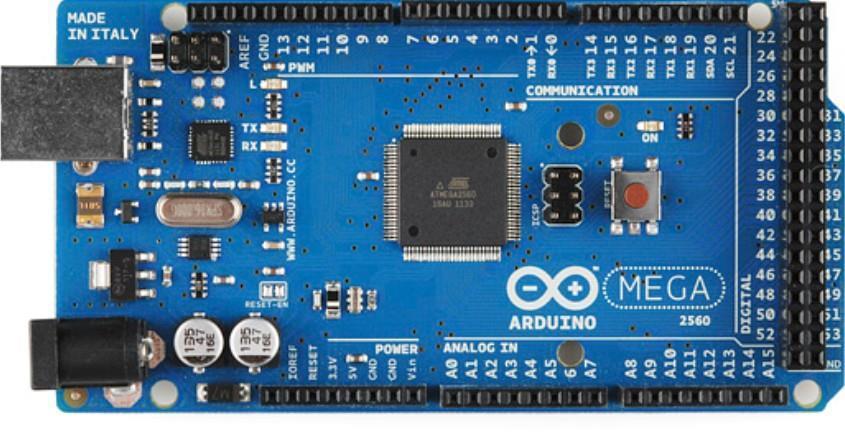


Figure Arduino Mega2560 board

## SENSORS

“In the broadest definition, a sensor is a device, module, or subsystem whose purpose is to detect events or changes in its environment and send the information to other electronics, frequently a computer processor. A sensor is always used with other electronics, whether as simple as a light or as complex as a computer.

Sensors are used in everyday objects such as touch-sensitive elevator buttons (tactile sensor) and lamps which dim or brighten by touching the base, besides innumerable applications of which most people are never aware.

With advances in micromachinery and easy-to-use microcontroller platforms, the uses of sensors have expanded beyond the traditional fields of temperature, pressure or flow measurement,[1] for example into MARG sensors. Moreover, analog sensors such as potentiometers and force-sensing resistors are still widely used. Applications include manufacturing and machinery, airplanes and aerospace, cars, medicine, robotics and many other aspects of our day-to-day life.

A sensor's sensitivity indicates how much the sensor's output changes when the input quantity being measured changes. For instance, if the mercury in a thermometer moves 1 cm when the temperature changes by 1 °C, the sensitivity is 1 cm/°C (it is basically the slope Dy/Dx assuming a linear characteristic). Some sensors can also affect what they measure; for instance, a room temperature thermometer inserted into a hot cup of liquid cools the liquid while the liquid heats the thermometer. Sensors are usually designed to have a small effect on what is measured; making the sensor smaller often improves this and may introduce other advantages.[2] Technological progress allows more and more sensors to be manufactured on a microscopic scale as microsensors using MEMS technology. In most cases, a microsensor reaches a significantly higher speed and sensitivity compared with macroscopic approaches.[3][4]”[ https://en.wikipedia.org/wiki/Sensor]

1. Bennett, S. (1993). A History of Control Engineering 1930–1955. London: Peter Peregrinus Ltd. on behalf of the Institution of Electrical Engineers. [*ISBN*](https://en.wikipedia.org/wiki/International_Standard_Book_Number) [*0-86341-280-7*](https://en.wikipedia.org/wiki/Special:BookSources/0-86341-280-7)<The source states "controls" rather than "sensors", so its applicability is assumed. Many units are derived from the basic measurements to which it refers, such as a liquid's level measured by a differential pressure sensor.>
2. [**Jump up^**](https://en.wikipedia.org/wiki/Sensor#cite_ref-2) Jihong Yan (2015). [*Machinery Prognostics and Prognosis Oriented Maintenance Management*](https://books.google.com/books?id=LbzlBQAAQBAJ&pg=PA107). Wiley & Sons Singapore Pte. Ltd. p. 107.
3. [**Jump up^**](https://en.wikipedia.org/wiki/Sensor#cite_ref-3) Jihong Yan (2015). [*Machinery Prognostics and Prognosis Oriented Maintenance Management*](https://books.google.com/books?id=LbzlBQAAQBAJ&pg=PA108). Wiley & Sons Singapore Pte. Ltd. p. 108.
4. [**Jump up^**](https://en.wikipedia.org/wiki/Sensor#cite_ref-4) Ganesh Kumar. [*Modern General Knowledge*](https://books.google.com/books?id=DbnFSqKSVb0C&pg=PA194). Upkar Prakashan. p. 194. [*ISBN*](https://en.wikipedia.org/wiki/International_Standard_Book_Number) [*81-7482-180-5*](https://en.wikipedia.org/wiki/Special:BookSources/81-7482-180-5).

“Classification of Sensors

There are several classifications of sensors made by different authors and experts. Some are very simple and some are very complex. The following classification of sensors may already be used by an expert in the subject but this is a very simple classification of sensors.

In the first classification of the sensors, they are divided in to Active and Passive. Active Sensors are those which require an external excitation signal or a power signal.

Passive Sensors, on the other hand, do not require any external power signal and directly generates output response.

The other type of classification is based on the means of detection used in the sensor. Some of the means of detection are Electric, Biological, Chemical, Radioactive etc.

The next classification is based on conversion phenomenon i.e. the input and the output. Some of the common conversion phenomena are Photoelectric, Thermoelectric, Electrochemical, Electromagnetic, Thermooptic, etc.

The final classification of the sensors are Analog and Digital Sensors. Analog Sensors produce an analog output i.e. a continuous output signal with respect to the quantity being measured.

Digital Sensors, in contrast to Analog Sensors, work with discrete or digital data. The data in digital sensors, which is used for conversion and transmission, is digital in nature.

Different Types of Sensors

The following is a list of different types of sensors that are commonly used in various applications. All these sensors are used for measuring one of the physical properties like Temperature, Resistance, Capacitance, Conduction, Heat Transfer etc.

* Temperature Sensor
* Proximity Sensor
* Accelerometer
* IR Sensor (Infrared Sensor)
* Pressure Sensor
* Light Sensor
* Ultrasonic Sensor
* Smoke, Gas and Alcohol Sensor
* Touch Sensor
* Color Sensor
* Humidity Sensor
* Tilt Sensor
* Flow and Level Sensor” [https://www.electronicshub.org/different-types-sensors/]

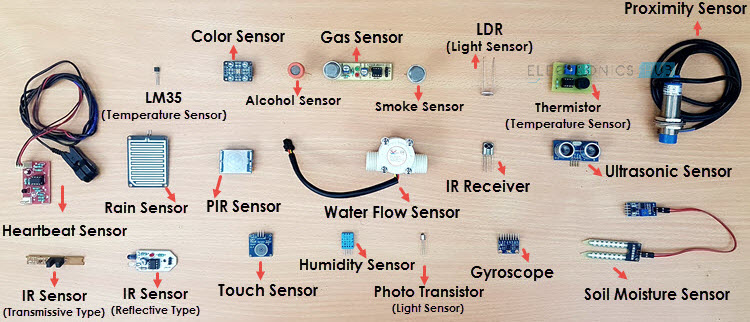


Figure Different types of sensors

### Temperature Sensor

One of the most common and most popular sensor is the Temperature Sensor. A Temperature Sensor, as the name suggests, senses the temperature i.e. it measures the changes in the temperature.

In a Temperature Sensor, the changes in the Temperature correspond to change in its physical property like resistance or voltage.

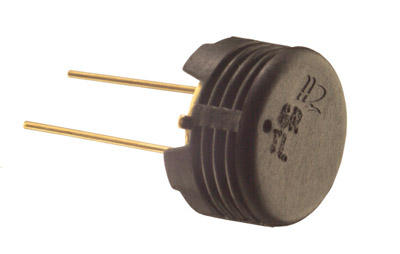
There are different types of Temperature Sensors like Temperature Sensor ICs (like LM35), Thermistors, Thermocouples, RTD (Resistive Temperature Devices), etc.

Temperature Sensors are used everywhere like computers, mobile

phones, automobiles, air conditioning systems, industries etc.

Figure Temperature Sensor

Humidity Sensors

“Humidity is the presence of water in air. The amount of water vapor in air can affect human comfort as well as many manufacturing processes in industries. The presence of water vapor also influences various physical, chemical, and biological processes. Humidity measurement in industries is critical because it may affect the business cost of the product and the health and safety of the personnel. Hence, **humidity sensing** is very important, especially in the control systems for industrial processes and human comfort.

F Figure Humidity Sensor

Controlling or monitoring humidity is of paramount importance in many industrial & domestic applications. In semiconductor industry, humidity or moisture levels needs to be properly controlled & monitored during wafer processing. In medical applications, humidity control is required for respiratory equipment, sterilizers, incubators, pharmaceutical processing, and biological products. Humidity control is also necessary in chemical gas purification, dryers, ovens, film desiccation, paper and textile production, and food processing. In agriculture, measurement of humidity is important for plantation protection (dew prevention), soil moisture monitoring, etc. For domestic applications, humidity control is required for living environment in buildings, cooking control for microwave ovens, etc. In all such applications and many others, humidity sensors are employed to provide an indication of the moisture levels in the environment.” [https://www.engineersgarage.com/articles/humidity-sensor]

## Liquid-crystal display

### Definition

“LCD (liquid crystal display) is the technology used for displays in notebook and other smaller computers. Like light-emitting diode (LED) and gas-plasma technologies, LCDs allow displays to be much thinner than cathode ray tube (CRT) technology. LCDs consume much less power than LED and gas-display displays because they work on the principle of blocking light rather than emitting it.

An LCD is made with either a passive matrix or an active matrix display display grid. The active matrix LCD is also known as a thin film transistor (TFT) display. The passive matrix LCD has a grid of conductors with pixels located at each intersection in the grid. A current is sent across two conductors on the grid to control the light for any pixel. An active matrix has a transistor located at each pixel intersection, requiring less current to control the luminance of a pixel.”[ https://whatis.techtarget.com/definition/LCD-liquid-crystal-display]

### LCD 16 x 2

“ A 16x2 LCD means it can display 16 characters per line and there are 2 such lines. In this LCD each character is displayed in 5x7 pixel matrix. This LCD has two registers, namely, Command and Data.

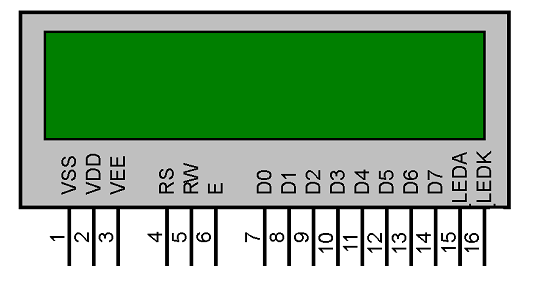
The command register stores the command instructions given to the LCD. A command is an instruction given to LCD to do a predefined task like initializing it, clearing its screen, setting the cursor position, controlling display etc. The data register stores the data to be displayed on the LCD. The data is the ASCII value of the character to be displayed on the LCD.” [https://www.engineersgarage.com/electronic-components/16x2-lcd-module-datasheet]

Figure LCD 16 x 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sr. No** | **Pin No.** | **Pin Name** | **Pin Type** | **Pin Description** | **Pin Connection** |
| 1 | Pin 1 | Ground | Source Pin | This is a ground pin of LCD | Connected to the ground of the MCU/ Power source |
| 2 | Pin 2 | VCC | Source Pin | This is the supply voltage pin of LCD | Connected to the supply pin of Power source |
| 3 | Pin 3 | V0/VEE | Control Pin | Adjusts the contrast of the LCD. | Connected to a variable POT that can source 0-5V |
| 4 | Pin 4 | Register Select | Control Pin | Toggles between Command/Data Register | Connected to a MCU pin and gets either 0 or 1.  0 -> Command Mode  1-> Data Mode |
| 5 | Pin 5 | Read/Write | Control Pin | Toggles the LCD between Read/Write Operation | Connected to a MCU pin and gets either 0 or 1.  0 -> Write Operation  1-> Read Operation |
| 6 | Pin 6 | Enable | Control Pin | Must be held high to perform Read/Write Operation | Connected to MCU and always held high. |
| 7 | Pin 7-14 | Data Bits (0-7) | Data/Command Pin | Pins used to send Command or data to the LCD. | In 4-Wire Mode  Only 4 pins (0-3) is connected to MCU  In 8-Wire Mode  All 8 pins(0-7) are connected to MCU |
| 8 | Pin 15 | LED Positive | LED Pin | Normal LED like operation to illuminate the LCD | Connected to +5V |
| 9 | Pin 16 | LED Negative | LED Pin | Normal LED like operation to illuminate the LCD connected with GND. | Connected to ground |

## Power supply

In this project all the components are connected to the laptop.

# SOFTWARE COMPONENTS USED

## Microsoft Visual Studio



“Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs, as well as web sites, web apps, web services and mobile apps. Visual Studio uses Microsoft software development platforms such as Windows API, Windows Forms, Windows Presentation Foundation, Windows Store and Microsoft Silverlight. It can produce both native code and managed code.

Figure 11 Microsoft Visual Studio

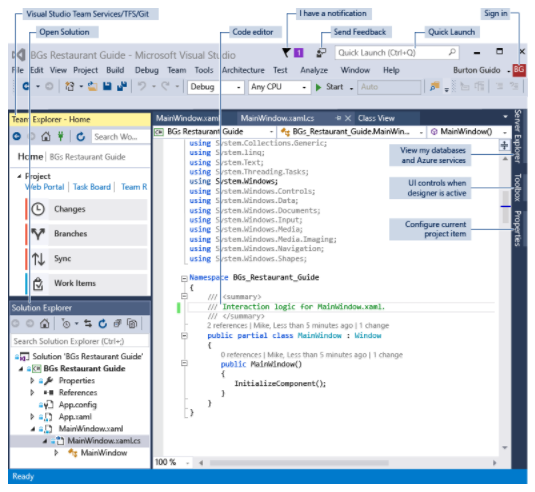
Visual Studio supports 36 different programming languages and allows the code editor and debugger to support (to varying degrees) nearly any programming language, provided a language-specific service exists. Built-in languages include C, C++, C++/CLI, Visual Basic .NET, C#, F#,JavaScript, TypeScript, XML, XSLT, HTML and CSS. Support for other languages such as Python, Ruby, Node.js, and M among others is available via plug-ins. Java (and J#) were supported in the past. “[https://en.wikipedia.org/wiki/Microsoft\_Visual\_Studio]

Figure Microsoft Visual Studio IDE basics

[https://msdn.microsoft.com/en-us/library/dn762121.aspx]

For the implementation of the desktop application I used Microsoft Visual Studio 2017a and the code was written in C#.

## Microsoft SQL Server

“Microsoft SQL Server is a relational database management system developed by Microsoft. As a database server, it is a software product with the primary function of storing and retrieving data as requested by other software applications—which may run either on the same computer or on another computer across a network (including the Internet).”[ <https://en.wikipedia.org/wiki/Microsoft_SQL_Server>]

“A database server is a server which houses a database application that provides database services to other computer programs or to computers, as defined by the client–server model. Database management systems frequently provide database-server functionality, and some database management systems (DBMSs) (such as MySQL) rely exclusively on the client–server model for database access (while others e.g. SQLite are meant for using as an embedded database).

Users access a database server either through a "front end" running on the user's computer – which displays requested data – or through the "back end", which runs on the server and handles tasks such as data analysis and storage.

In a master-slave model, database master servers are central and primary locations of data while database slave servers are synchronized backups of the master acting as proxies.

Most database applications respond to a query language. Each database understands its query language and converts each submitted query to server-readable form and executes it to retrieve results.”

[https://en.wikipedia.org/wiki/Database\_server]

The database management system that I used in order to be able to implement the database was SQL Server 2017.

Figure 13 Microsoft SQL Server

## SQL Server Management Studio

“SQL Server Management Studio (SSMS) is an integrated environment for managing any SQL infrastructure. Use SSMS to access, configure, manage, administer, and develop all components of SQL Server, Azure SQL Database, and SQL Data Warehouse. SSMS provides a single comprehensive utility that combines a broad group of graphical tools with a number of rich script editors to provide access to SQL Server for developers and database administrators of all skill levels.”[ <https://docs.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms>]

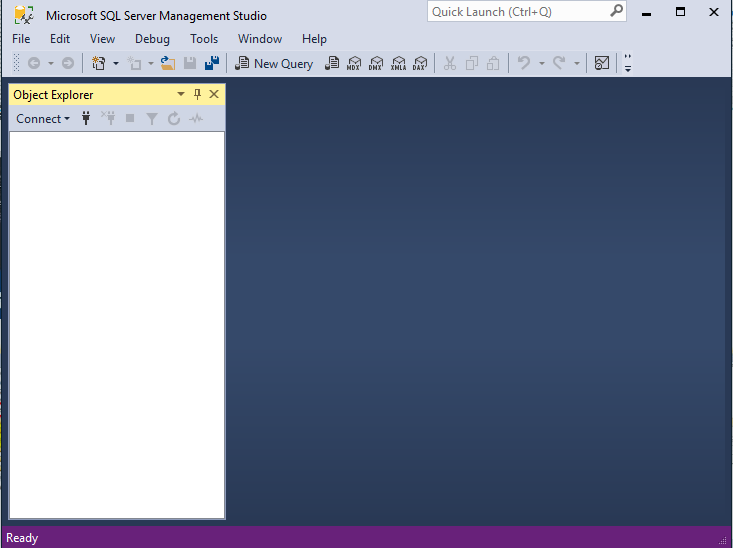


Figure 14 Microsoft SQL Server Management Studio

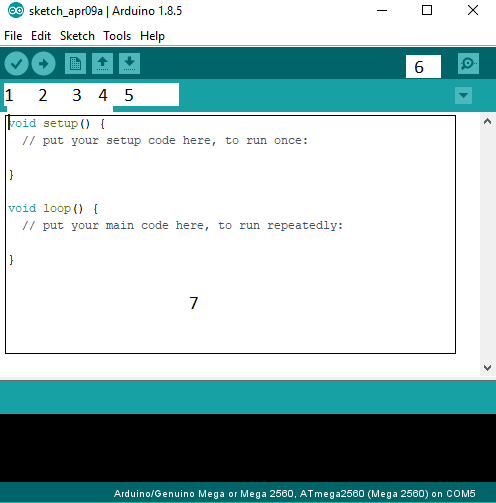
“SQL Server Management Studio provides features for managing objects in instances of the Database Engine, Analysis Services, Integration Services, and Reporting Services.

Benefits of Object Explorer

Object Explorer provides a hierarchical user interface to view and manage the objects in each instance of SQL Server. The Object Explorer Details pane presents a tabular view of instance objects, and the capability to search for specific objects. The capabilities of Object Explorer vary slightly depending on the type of server, but generally include the development features for databases, and management features for all server types.”[ <https://docs.microsoft.com/en-us/sql/ssms/object/object-explorer>]

## Arduino IDE

“The Arduino integrated development environment (IDE) is a cross-platform application (for Windows, macOS, Linux) that is written in the programming language Java. It originated from the IDE for the languages Processing and Wiring. It includes a code editor with features such as text cutting and pasting, searching and replacing text, automatic indenting, brace matching, and syntax highlighting, and provides simple one-click mechanisms to compile and upload programs to an Arduino board. It also contains a message area, a text console, a toolbar with buttons for common functions and a hierarchy of operation menus.

The Arduino IDE supports the languages C and C++ using special rules of code structuring. The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub main() into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution. The Arduino IDE employs the program avrdude to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware.”[ [https://en.wikipedia.org/wiki/Arduino#IDE](https://en.wikipedia.org/wiki/Arduino" \l "IDE)]

|  |  |
| --- | --- |
| 1 | Verify |
| 2 | Upload |
| 3 | New |
| 4 | Open |
| 5 | Save |
| 6 | Serial Monitor |
| 7 | Coding Area |

Figure Arduino IDE

## Fritzing IDE

Fritzing is an open source initiative to develop amateur or hobby CAD software for the design of electronics hardware, to support designers and artists ready to move from experimenting with a prototype to building a more permanent circuit. It was developed at the University of Applied Sciences of Potsdam.

The software is created in the spirit of the Processing programming language and the Arduino microcontroller and allows a designer, artist, researcher, or hobbyist to document their Arduino-based prototype and create a PCB layout for manufacturing. The associated website helps users share and discuss drafts and experiences as well as to reduce manufacturing costs.

Fritzing can be seen as an electronic design automation (EDA) tool for non-engineers: the input metaphor is inspired by the environment of designers (the breadboard-based prototype), while the output is focused on accessible means of production. As of December 2, 2014 Fritzing has made a code view option, where one can modify code and upload it directly to an Arduino device.

Figure Fritzing Logo

# HARDWARE ARHITECTURE

# SOFTWARE ARHITECTURE

## Namespaces used

### System.IO.Ports

“The System.IO.Ports namespace contains classes for controlling serial ports. The most important class, SerialPort, provides a framework for synchronous and event-driven I/O, access to pin and break states, and access to serial driver properties. It can be used to wrap a Stream objects, allowing the serial port to be accessed by classes that use streams.”[ <https://msdn.microsoft.com/en-us/library/system.io.ports(v=vs.90).aspx>]

From this namespace SerialPort class was used in order to be able to communicate with the Arduino board.

“SerialPort Class syntax:

C# public class SerialPort : Component

Constructors used:

* SerialPort(): Initializes a new instance of the SerialPort class.
* SerialPort(String, Int32, Parity, Int32, StopBits): Initializes a new instance of the SerialPort class using the specified port name, baud rate, parity bit, data bits, and stop bit.

Methods used:

* GetPortNames(): Gets an array of serial port names for the current computer.
* Open(): Opens a new serial port connection.
* Close(): Closes the port connection, sets the IsOpen property to false, and disposes of the internal Stream object.
* Write(String): Writes the specified string to the serial port. “ [https://msdn.microsoft.com/en-us/library/system.io.ports.serialport(v=vs.110).aspx]

### System.Xml

The System.Xml namespace provides standards-based support for processing XML.[ https://msdn.microsoft.com/library/system.xml.aspx]

“What is XML?

* XML stands for eXtensible Markup Language
* XML is a markup language much like HTML
* XML was designed to store and transport data
* XML was designed to be self-descriptive
* XML is a World Wide Web Consortium Recommendation”[ <https://www.w3schools.com/xml/xml_whatis.asp>]

“The design goals of XML emphasize simplicity, generality, and usability across the Internet. It is a textual data format with strong support via Unicode for different human languages. Although the design of XML focuses on documents, the language is widely used for the representation of arbitrary data structures such as those used in web services.

Several schema systems exist to aid in the definition of XML-based languages, while programmers have developed many application programming interfaces (APIs) to aid the processing of XML data.

Hundreds of document formats using XML syntax have been developed, including RSS, Atom, SOAP, SVG, and XHTML. XML-based formats have become the default for many office-productivity tools, including Microsoft Office (Office Open XML), OpenOffice.org and LibreOffice (OpenDocument), and Apple's iWork. XML has also provided the base language for communication protocols such as XMPP. Applications for the Microsoft .NET Framework use XML files for configuration. Apple has an implementation of a registry based on XML.”[ <https://en.wikipedia.org/wiki/XML>]

The class used in this application is XmlReader.

“XmlReader represents a reader that provides fast, noncached, forward-only access to XML data.

XMLReader syntax:

C#: public abstract class XmlReader : IDisposable

Constructor used:

* XmlReader(): Initializes a new instance of the XmlReader class.

Method used:

* Create(String): Creates a new XmlReader instance with specified URI.”[ https://msdn.microsoft.com/en-us/library/system.xml.xmlreader.aspx]

### System.ServiceModel.Syndication

In order to be able to get the RSS the SyndicationFeed class was used.

What is a RSS?

“RSS (Rich Site Summary; originally RDF Site Summary; often called Really Simple Syndication) is a type of web feed which allows users to access updates to online content in a standardized, computer-readable format. These feeds can, for example, allow a user to keep track of many different websites in a single news aggregator. The news aggregator will automatically check the RSS feed for new content, allowing the content to be automatically passed from website to website or from website to user. This passing of content is called web syndication. Websites usually use RSS feeds to publish frequently updated information, such as blog entries, news headlines, audio, video. An RSS document (called "feed", "web feed", or "channel") includes full or summarized text, and metadata, like publishing date and author's name.

A standard XML file format ensures compatibility with many different machines/programs. RSS feeds also benefit users who want to receive timely updates from favorite websites or to aggregate data from many sites.

Subscribing to a website RSS removes the need for the user to manually check the website for new content. Instead, their browser constantly monitors the site and informs the user of any updates. The browser can also be commanded to automatically download the new data for the user.

RSS feed data is presented to users using software called a news aggregator. This aggregator can be built into a website, installed on a desktop computer, or installed on a mobile device. Users subscribe to feeds either by entering a feed's URI into the reader or by clicking on the browser's feed icon. The RSS reader checks the user's feeds regularly for new information and can automatically download it, if that function is enabled. The reader also provides a user interface.” [https://en.wikipedia.org/wiki/RSS]

“SyndicationFeed Class syntax:

C#: public class SyndicationFeed

Constructor used:

* SyndicationFeed(): Initializes a new instance of the SyndicationFeed class.

Methods used:

* Load(XmlReader): Loads a syndication feed from the specified XML reader.

When serialized to Atom 1.0, a SyndicationFeed instance is written to a <**feed**> element. The following table shows how each property defined in the SyndicationFeed class is serialized to Atom 1.0.

|  |  |
| --- | --- |
| **SyndicationFeed property** | **Serialized form** |
| **AttributeExtensions** | An attribute in the <**feed**> element for each attribute in the collection. |
| **Authors** | An <**author**> element for each SyndicationPerson in the collection. |
| **Categories** | A <**category**> element for each SyndicationCategory in the collection. |
| **Contributors** | A <**contributor**> element for each SyndicationPerson in the collection. |
| **Copyright** | A <**rights**> element. |
| **Description** | A <**subtitle**> element. |
| **ElementExtensions** | Each element in the collection is written within the <**feed**> element. |
| **Generator** | A <**generator**> element. |
| **Id** | An <**id**>element. |
| **ImageUri** | A <**logo**> element. |
| **Items** | An <**entry**> element for each SyndicationItem in the collection. |
| **Language** | Not serialized. |
| **LastUpdatedTime** | An <**updated**> element. |
| **Links** | A <**link**> element for each SyndicationLink in the collection. |
| **Title** | A <**title**> element. |

When serialized to RSS 2.0, a SyndicationFeed instance is written to an <**rss**> element. The following

table shows how each property defined in the SyndicationFeed class is serialized to RSS 2.0.

|  |  |
| --- | --- |
| **SyndicationFeed property** | **Serialized form** |
| **AttributeExtensions** | An attribute in the <**channel**> element for each attribute in the collection. |
| **Authors** | A <**managingEditor**> element if only one SyndicationPerson is in the collection; otherwise, an <**a10:author**> element for each SyndicationPerson in the collection. |
| **Categories** | A <**category**> element for each SyndicationCategory in the collection. |
| **Contributors** | An <**a10:contributor**> element for each SyndicationPerson in the collection. |
| **Copyright** | A <**copyright**> element. |
| **Description** | A <**description**> element. |
| **ElementExtensions** | Each element in the collection is written within the <**channel**> element. |
| **Generator** | A <**generator**> element. |
| **Id** | An <**a10:id**> element. |
| **ImageUri** | An <**image**> element. |
| **Items** | An <**item**> element for each SyndicationItem in the collection. |
| **Language** | A <**language**> element. |
| **LastUpdatedTime** | A <**lastBuildDate**> element. |
| **Links** | An <**a10:link**> element for each SyndicationLink in the collection. |
| **Title** | A <**title**> element. |

### System.Windows.Forms

“The System.Windows.Forms namespace contains classes for creating Windows-based applications that take full advantage of the rich user interface features available in the Microsoft Windows operating system.”[ <https://msdn.microsoft.com/en-us/library/system.windows.forms(v=vs.110).aspx>]

MessageBox Class was used in order to present a message to the user.

“MessageBox Class syntax:

C#: public class MessageBox

Method used:

* Show(String): Displays a message box with specified text.” [https://msdn.microsoft.com/en-us/library/system.windows.forms.messagebox(v=vs.110).aspx]

### System.Data.SqlClient

“The System.Data.SqlClient namespace is the.NET Framework Data Provider for SQL Server.

Classes used:

* SqlConnection: Represents an open connection to a SQL Server database. This class cannot be inherited.
* SqlCommand: Represents a Transact-SQL statement or stored procedure to execute against a SQL Server database. This class cannot be inherited.
* SqlDataAdapter: Represents a set of data commands and a database connection that are used to fill the DataSet and update a SQL Server database. This class cannot be inherited.

”[ https://msdn.microsoft.com/en-us/library/system.data.sqlclient(v=vs.110).aspx]

“SqlConnection Class syntax:

C#: public sealed class SqlConnection : DbConnection, ICloneable

Constructor used:

* SqlConnection(String): Initializes a new instance of the SqlConnection class when given a string that contains the connection string.

Method used:

* Open(): Opens a database connection with the property settings specified by the ConnectionString.(Overrides DbConnection.Open().)” [https://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlconnection(v=vs.110).aspx]

“SqlCommand syntax:

C#: public sealed class SqlCommand : DbCommand, ICloneable

Constructor used:

* SqlCommand(String, SqlConnection): Initializes a new instance of the SqlCommand class with the text of the query and a SqlConnection.”[ <https://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand(v=vs.110).aspx>]

“SqlDataAdapter syntax

C#: public sealed class SqlDataAdapter : DbDataAdapter, IDbDataAdapter, IDataAdapter, ICloneable

Constructor used:

* SqlDataAdapter(String, SqlConnection): Initializes a new instance of the SqlDataAdapter class with a SelectCommand and a SqlConnection object.”[ <https://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldataadapter(v=vs.110).aspx>]

## Database

### Entity Framework

In order to access the database in the desktop application a persistence framework is needed.

With the help of the persistence framework the objects from the application will be load and saved in the database.

“A persistence framework is a middleware that assists and automates the storage of program data into databases, especially relational databases. It acts as a layer of abstraction between the application and the database, typically bridging any conceptual differences between the two.”[ <https://en.wikipedia.org/wiki/Persistence_framework>]

“Entity Framework (EF) is an open source object-relational mapping (ORM) framework for ADO.NET.

The Entity Framework is a set of technologies in ADO.NET that support the development of data-oriented software applications. Architects and developers of data-oriented applications have typically struggled with the need to achieve two very different objectives. They must model the entities, relationships, and logic of the business problems they are solving, and they must also work with the data engines used to store and retrieve the data. The data may span multiple storage systems, each with its own protocols; even applications that work with a single storage system must balance the requirements of the storage system against the requirements of writing efficient and maintainable application code.

The Entity Framework enables developers to work with data in the form of domain-specific objects and properties, such as customers and customer addresses, without having to concern themselves with the underlying database tables and columns where this data is stored. With the Entity Framework, developers can work at a higher level of abstraction when they deal with data, and can create and maintain data-oriented applications with less code than in traditional applications.”[ <https://en.wikipedia.org/wiki/Entity_Framework>]

Entity Framework workflows

* Database-First
  + - * The tables are design
      * EF generates the domain classes

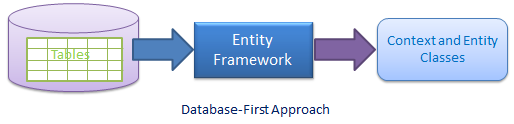


Figure Database first

* Model-First
  + An UML diagram is created
  + EF generates domain classes and database

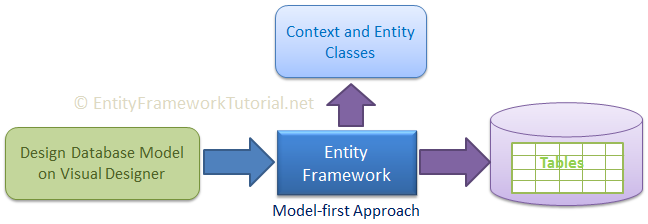


Figure Model first

* Code-First
  + The domain classes are created
  + EF generates database tables

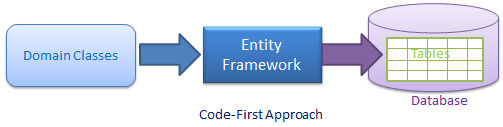


Figure 19 Code First

In order to create the database needed for this project, code-first approach was used.

“The development workflow in the code-first approach would be: Create or modify domain classes -> configure these domain classes using Fluent-API or data annotation attributes -> Create or update the database schema using automated migration or code-based migration.”[ http://www.entityframeworktutorial.net/code-first/what-is-code-first.aspx]

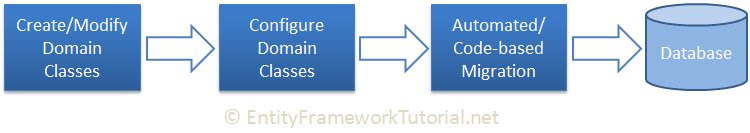
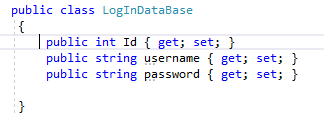
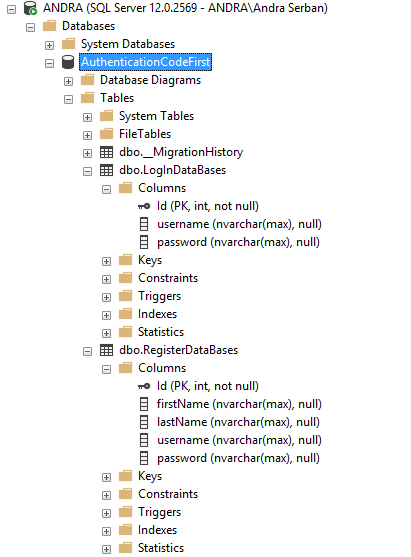
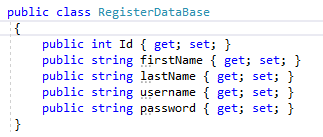


Figure 20 Code-First Workflow

### Database Structure





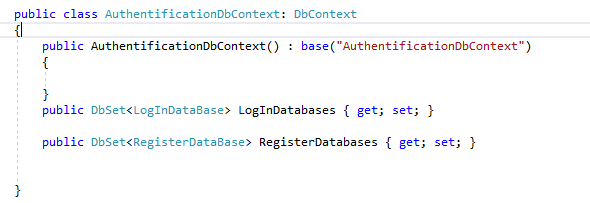


Figure Mapping between code and database

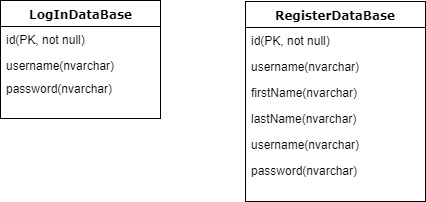


Figure Database UML diagram

In order to be able to access the main application, a log in is necessary.

As can be seen in the diagram the log in table as a primary key – the Id, a username and a password.

Moreover if the employee does not have an account then it has to create one by accessing the register command.

The register table has a primary key – the Id, the first and last name of the employee, a username and a password.

C# application

The C# application is the one that interacts with the user.

In this section I will explain the classes, the forms and how the functionality of the desktop application was build.

### Log In

In order to have access to the main application the user has to log in based on a username and a password.

Firstly on the screen the log in interface appears and require a username and a password.

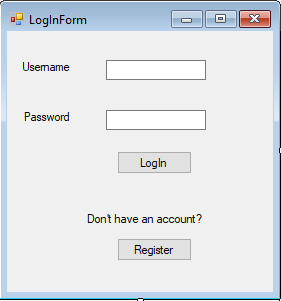


Figure LogIn interface

I used a label and a textbox for username and password.

On the submit button is implemented an event functionality which triggers on click. After the connection with the database is establish is checked of the information needed is provided.

Both textboxes must conation information. If this the textboxes do not contain any information a message box appears on the screen with the fallowing string: “Please provide username and password” .

Below are illustrated the cases that validates the check of the username and password textboxes:

* Case1: Neither the username nor the password has been entered

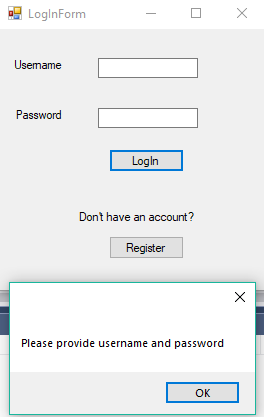


Figure Provide username and password: case1

* Case2: Only the username is provided

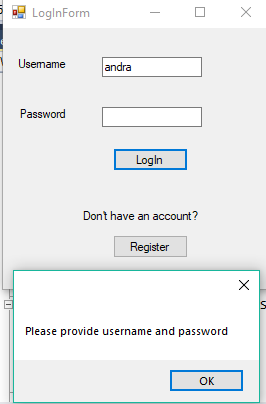


Figure Provide username and password: case2

* Case3: Only the password is entered.

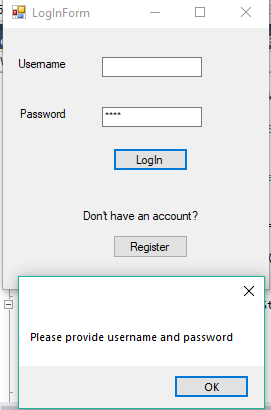


Figure Provide username and password: case3

Moreover if the username and the password are not valid another message box appears with the fallowing string: “Invalid username or password”.

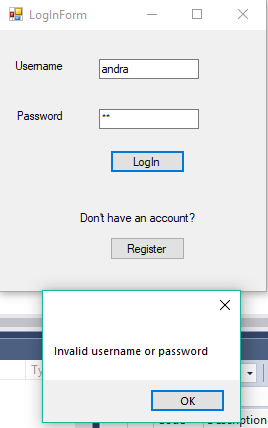


Figure Invalid username or password

### Register

As the label from the login form suggests if the employee does not have an account then it has to register.

The register button contains an event which triggers on click. After the register button is trigger a new form is displayed, the register form.

The register forms contains the fallowing labels and textboxes:

* First name
* Last name
* Username
* Password
* Confirm password

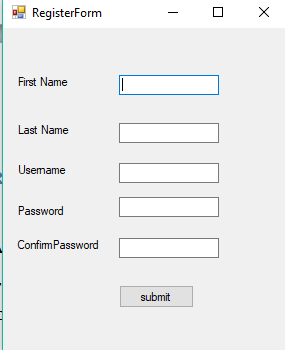


Figure Register Interface

First of all the connection with the database is made. If the connection is done successfully then the information introduced by the user is inserted in the database tables using a SQL Command.

There are two labels that have to match: password and confirm password. A message box with the fallowing string will the fallowing string: “Passwords do not match” will appear if there is a mismatch between the two textboxes mentioned above.

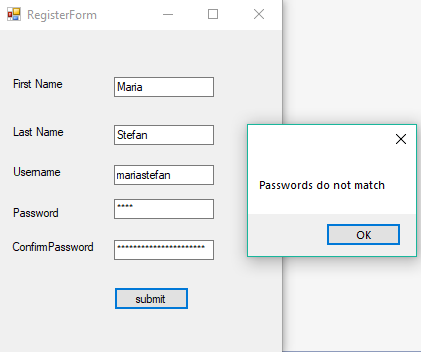
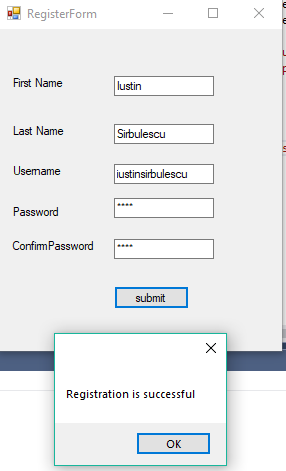


Figure Passwords do not match



The registration is done successfully is this two labels match and a message box will announce the user: “Registration is successful

Figure Registration is successful

Main Form

How the desktop application communicates with Arduino?

For this project I have used a communication protocol. The protocol that I used in order to make possible the communication between the desktop application and the board consists in 4 commands.

Each command has at the beginning the # character and at the end is followed by the \n symbol. With the help of this symbol the Arduino board knows that command is complete and it can start acting. Each command can be followed by some data and has is 4 characters long.

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **COMMAND** | **DATA** | **\n** |

Figure Command Structure

The fallowing commands were used:

* STAR (for start)
* STOP
* TEXT

STAR command: with the help of this command the desktop application tells Arduino to wait for inputs.

|  |  |  |
| --- | --- | --- |
| **#** | **STAR** | **\n** |

Figure STAR command

STOP command: by using this command Arduino disconnects from the desktop application.

|  |  |  |
| --- | --- | --- |
| **#** | **STOP** | **\n** |

Figure STOP command

TEXT command: this command is followed by the text that will be displayed on the LCD.

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **TEXT** | **TEXT TO DISPLAY** | **\n** |

Figure TEXT command

Moving forward let’s now talk about the RSS reader. Steps:

* For managing the web feed named Rich Site Summary XmlReader Class was used.” XmlReader represents a reader that provides fast, no cached, forward-only access to XML data.”[ <https://docs.microsoft.com/en-us/dotnet/api/system.xml.xmlreader?view=netframework-4.7.2>]
* The information is read from the url introduced in the textbox marked in the figure below.
* After that by calling the static Load(XmlReader) method, passing in the XmlReader from the textbox.
* SyndicationFeed feed = SyndicationFeed.Load(readXml): ”This invokes the service operation and populates a new SyndicationFeed with the formatter returned from the service operation.”[ <https://docs.microsoft.com/en-us/dotnet/framework/wcf/feature-details/how-to-create-a-basic-rss-feed>]

## Arduino application

The Arduino application is the one that will be upload on the board. This application is responsible with the pin configuration as well as read/write the pins.

The desktop application will send serial commands to the Arduino application, but will also receive data from the LCD and the sensors.

This application uses two functions, namely:

* Setup function;
* Loop function;

Arduino code steps for displaying the information from the C# apllication.

* At first the display is initialized.
* Then I wait for serial data.
* When I receive data on the serial port I save the data in the variable inputString until the \n character is received.
* When I receive this character I know that command is complete.
* If the TEXT command is received then I know that I have to print the string on the LCD display.
* The first five characters from the inputString variable, because that’s the command, I delete then last 2 character which is the \n symbol and I print the remaining characters on the display.

Arduino code steps for getting the humidity and temperature:

* Two variables are declared for humidity and temperature.
* The vales are read using a functions from the DHT library
* In order to be able to display the humidity and temperature the float variables are converted into others string variables.