TEI OF CENTRAL MACEDONIA - SERRES

M.Sc. in Communication and Information Systems

A microcontroller-based system for multi sensor monitoring and messaging via GSM network

Bachelor thesis

Angelakis Vaios

Supervisor: **Kazarlis S.**

Serres, November 2015

Introduction

Nowadays where the human need for quality but cheap farm products is essential.

The AgriArduino is a system inspecting sensors.

From the side of the farmers attempt to lower production costs but also improve product quality can be achieved using such systems. The AgriArduino is essentially a tool that updates directly the producers when there are risks to the farm. Recording the circumstances as well as what natural resources is available. In Greece, where the scale of each farmer is small and dispersed but also uneven-technology can help farmers to become more competitive.

CONTENTS **Pages**

The benefits of use IT in Agriculture

Given the challenges, the arrival of information communication technology (ICT) is well timed. The benefits of the green revolution greatly improved agricultural productivity.
However, there is a demonstrable need for a new revolution that will bring lower prices for consumers (through reduced waste and more-efficient supply chain management), contribute to “smart” agriculture, and incentivize farmers (for example, through higher income) to increase their production.

Public and private sector actors have long been on the search for effective solutions to address both the long- and short-term challenges in agriculture, including how to answer the abundant information needs of farmers. ICT is one of these solutions, and has recently unleashed incredible potential to improve agriculture in developing countries specifically. Technology has taken an enormous leap beyond the costly, bulky, energy-consuming equipment once available to the very few to store and analyze agricultural and scientific data.

With the booming mobile, wireless, and Internet industries, ICT has found a foothold even in poor smallholder farms and in their activities. The ability of ICTs to bring refreshed momentum to agriculture appears even more compelling in light of rising investments in agricultural research, the private sector’s strong interest in the development and spread of ICTs, and the upsurge of organizations committed to the agricultural development agenda.

But what exactly are ICTs? And can they really be useful and cost-effective for poor farmers with restricted access to capital, electricity, and infrastructure? First, an ICT is any device, tool, or application that permits the exchange or collection of data through interaction or transmission. ICT is an umbrella term that includes anything ranging from radio to satellite imagery to mobile phones or electronic money transfers. Second, these ICTs and others have gained traction even in impoverished regions. The increases in their affordability, accessibility, and adaptability have resulted in their use even within rural homesteads relying on agriculture. New, small devices (such as multifunctional mobile phones and nanotechnology for food safety), infrastructure (such as mobile telecommunications networks and cloud computing facilities), and especially applications (for example, that transfer money or track an item moving through a global supply chain) have proliferated.

Many of the questions asked by farmers (including questions on how to increase yields, access markets, and adapt to weather conditions) can now be answered faster, with greater ease, and increased accuracy. Many of the questions can also be answered with a dialogue—where farmers, experts, and government can select best solutions based on a diverse set of expertise and experience.

The types of ICT-enabled services that are useful to improving the capacity and livelihoods of poor smallholders are growing quickly. One of the best examples of these services is the use of mobile phones as a platform for exchanging information through short messaging services (SMS).
Reuters Market Light, for example, services over 200,000 smallholder subscribers in 10 different states in India for a cost of $1.50 per month. The farmers receive four to five messages per day on prices, commodities, and advisory services from a database with information on 150 crops and more than 1,000 markets. Preliminary evidence suggests that collectively, the service may have generated US$ 2–3 billion in income for farmers (Mehra 2010), while over 50 percent of them have reduced their spending on agriculture inputs.

ICT-enabled services often use multiple technologies to provide information. This model is being used to provide rural farmers localized (non-urban) forecasts so that they can prepare for weather-related events. In resource-constrained environments especially, providers use satellites or remote sensors (to gather temperature data), Internet (to store large amounts of data), and mobile phones (to disseminate temperature information to remote farmers cheaply)—to prevent crop losses and mitigate effects from natural adversities.

Other, more-specialized applications, such as software used for supply chain or financial management are also becoming more relevant in smallholder farming. Simple accounting software has allowed cooperatives to manage production, aggregation, and sales with increased accuracy.

The Malian Coprokazan, involved in shea butter production, began using solar-powered computers with keyboards adapted to the local language to file members' records electronically. Along with electronic administration, the coop plans to invest in Global Positioning System (GPS) technology to obtain certifications and use cameras and video as training materials to raise the quality of production. From 2006 to 2010 alone, the coop's membership grew from 400 to 1,000 producers

Chapter 1. Objective

The main objective is to create a general system for receiving data from an external device and immediately update - alert the user. Therefore the system

- measures the temperature and pressure of water
- alarm in extreme situations by calling or sms,
- inform about what happens in the external environment if received sms.

The system has two modes:
Control Water pressure

- when water pressure is low .
- when water pressure is high .

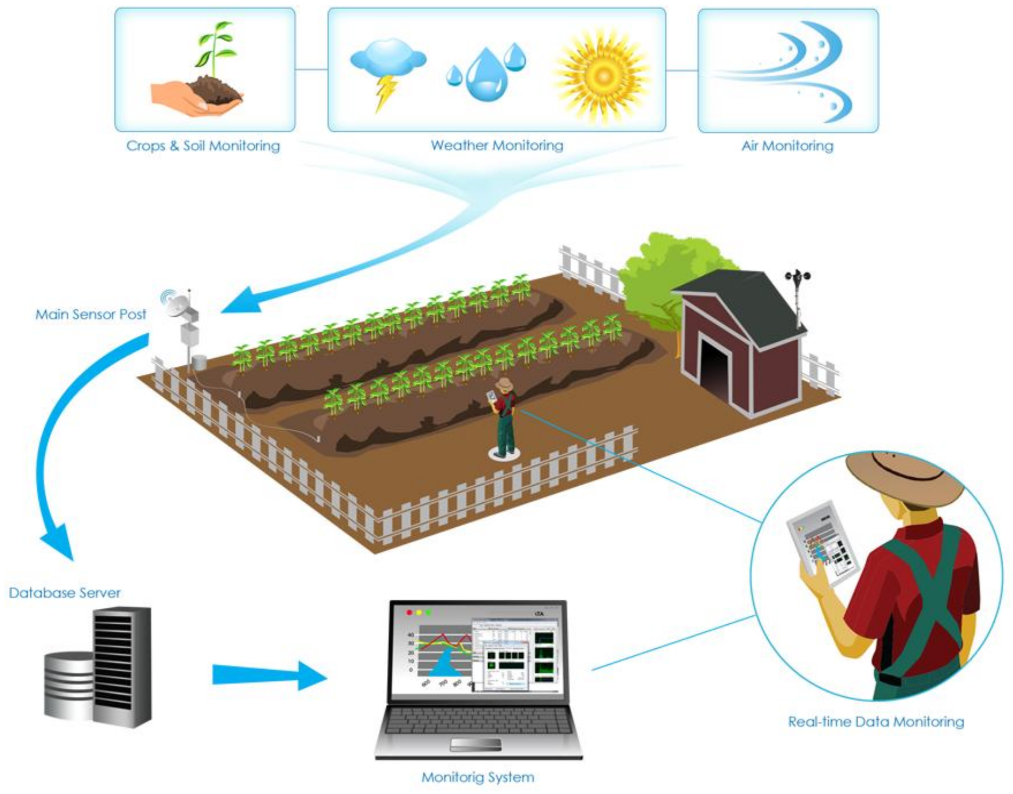
Control Temperature

- when the temperature to a normal level
- when it has surpassed the breakpoints

a. Precision Agriculture

In the past few years, new trends have emerged in the agricultural sector. Thanks to developments in the field of wireless sensor networks as well as miniaturization of the sensor boards, precision agriculture started emerging. Precision agriculture concentrates on providing the means for observing, assessing and controlling agricultural practices.

It covers a wide range of agricultural concerns from daily herd management through horticulture to field crop production. It concerns as well pre- and post-production aspects of agricultural enterprises.



A facet of precision agriculture concentrates on site-specific crop management. This encompasses different aspects, such as monitoring soil, crop and climate in a field generalizing the results to a complete parcel providing a decision support system (DSS) for delivering insight into possible treatments, field-wide or for specific parts of a field; and the means for taking differential action, for example, varying in real-time an operation such as fertilizer, lime and pesticide application, tillage, or sowing rate.

b. The AgriArduino

A microcontroller-based system for multi sensor monitoring and messaging via GSM network consists of hardware which is:

- Arduino Uno
- Gsm Shield
- Lcd Shield
- Lm35 (temp sensor)
- F83161.3 switch for water pressure

The AgriArduino (A microcontroller-based system for multi sensor monitoring and messaging via GSM network) enables us to accept calls when the temperature exceeds 35 degrees Celsius.
It knows when the water pressure is sufficient for watering.
Moreover we have the ability to send sms to inform the database located on a server.
The AgriArduino informs us about the data collected through either a website the use and application smartphone Android.

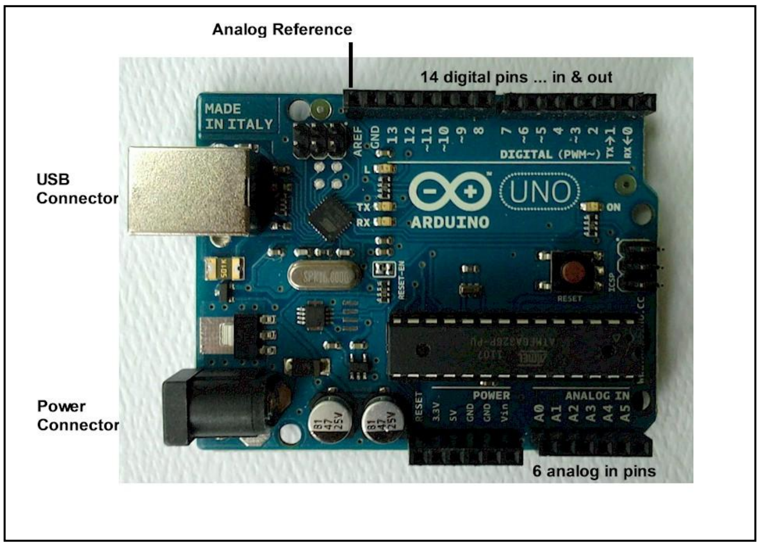
Chapter 2. Arduino

a. Hardware

Arduino Uno

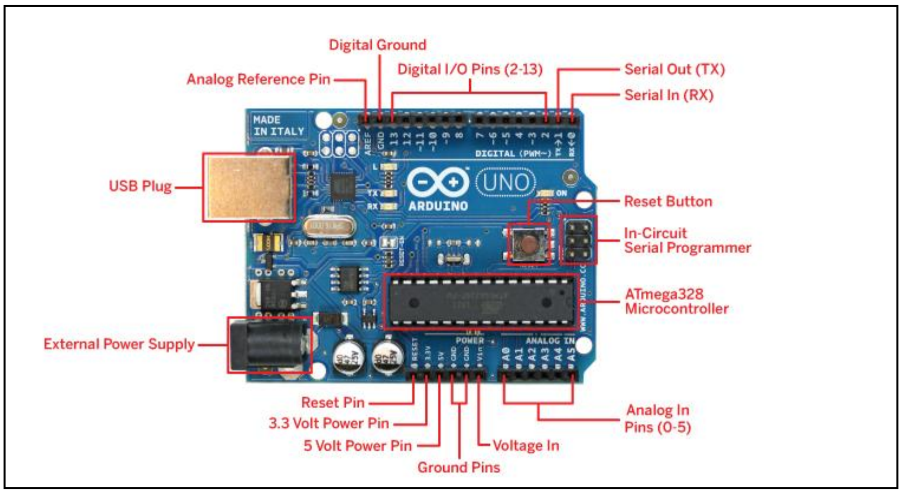
An Arduino board consists of an Atmel 8-, 16- or 32-bit AVR microcontroller with complementary components that facilitate programming and incorporation into other circuits. An important aspect of the Arduino is its standard connectors, which lets users connect the CPU board to a variety of interchangeable add-on modules known as *shields*. Some shields communicate with the Arduino board directly over various pins, but many shields are individually addressable via an I²C serial bus—so many shields can be stacked and used in parallel. Official Arduinos have used the megaAVRseries of chips, specifically the ATmega8, ATmega168, ATmega328, ATmega1280, and ATmega2560. A handful of other processors have been used by Arduino compatibles. Most boards include a 5 volt linear regulator and a 16 MHz crystal oscillator (or ceramic resonator in some variants), although some designs such as the LilyPad run at 8 MHz and dispense with the onboard voltage regulator due to specific form-factor restrictions. An Arduino's microcontroller is also pre-programmed with a boot loader that simplifies uploading of programs to the on-chip flash memory, compared with other devices that typically need an external programmer. This makes using an Arduino more straightforward by allowing the use of an ordinary computer as the programmer.

At a conceptual level, when using the Arduino software stack, all boards are programmed over an RS-232 serial connection, but the way this is implemented varies by hardware version. Serial Arduino boards contain a level shifter circuit to convert between RS-232-level and TTL-level signals. Current Arduino boards are programmed via USB, implemented using USB-to-serial adapter chips such as the FTDI FT232. Some variants, such as the Arduino Mini and the unofficial Boarduino, use a detachable USB-to-serial adapter board or cable, Bluetooth or other methods. (When used with traditional microcontroller tools instead of the Arduino IDE, standard AVR ISP programming is used.)



The Arduino board exposes most of the microcontroller's I/O pins for use by other circuits. The Diecimila, Duemilanove, and current Uno provide 14 digital I/O pins, six of which can produce pulse-width modulated signals, and six analog inputs, which can also be used as six digital I/O pins. These pins are on the top of the board, via female 0.10-inch (2.5 mm) headers. Several plug-in application shields are also commercially available. The Arduino Nano, and Arduino-compatible Bare Bones Board and Boarduino boards may provide male header pins on the underside of the board that can plug into solderless breadboards.

There are many Arduino-compatible and Arduino-derived boards. Some are functionally equivalent to an Arduino and can be used interchangeably. Many enhance the basic Arduino by adding output drivers, often for use in school-level education to simplify the construction of buggies and small robots. Others are electrically equivalent but change the form factor, sometimes retaining compatibility with shields, sometimes not. Some variants use completely different processors, with varying levels of compatibility.



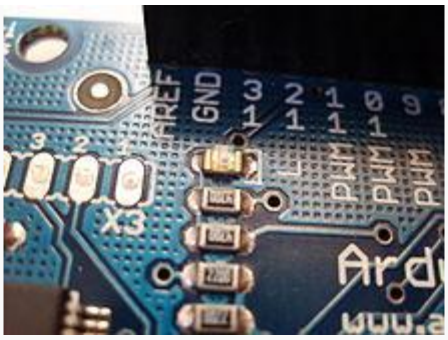
b. Software for Arduino

The Arduino integrated development environment (IDE) is a cross-platform application written in Java, and derives from the IDE for the Processing programming language and the Wiring projects. It is designed to introduce programming to artists and other newcomers unfamiliar with software development. It includes a code editor with features such as syntax highlighting, brace matching, and automatic indentation, and is also capable of compiling and uploading programs to the board with a single click. A program or code written for Arduino is called a "sketch".[16]

Arduino programs are written in C or C++. The Arduino IDE comes with a software library called "Wiring" from the original Wiring project, which makes many common input/output operations much easier. The users need only to define two functions to make an executable cyclic executive program:

- `setup()`: a function run once at the start of a program that can initialize settings
- `loop()`: a function called repeatedly until the board powers off

A typical first program for a microcontroller simply blinks an LED on and off. In the Arduino environment, the user might write a program like this:[17]



The integrated pin 13 LED

```
#define LED_PIN 13

void setup() {
   pinMode(LED_PIN, OUTPUT);      // Enable pin 13 for digital output
}

void loop() {
   digitalWrite(LED_PIN, HIGH);   // Turn on the LED
   delay(1000);                   // Wait one second (1000 milliseconds)
   digitalWrite(LED_PIN, LOW);    // Turn off the LED
   delay(1000);                   // Wait one second
}
```

Most Arduino boards contain an LED and a load resistor connected between the pin 13 and ground, which is a convenient feature for many simple tests.[17] The previous code would not be seen by a standard C++ compiler as a valid program, so when the user clicks the "Upload to I/O board" button in the IDE, a copy of the code is written to a temporary file with an extra include header at the top and a very simple main() function at the bottom, to make it a valid C++ program.

The Arduino IDE uses the GNU toolchain and AVR Libc to compile programs, and uses avrdude to upload programs to the board.

As the Arduino platform uses Atmel microcontrollers, Atmel's development environment, AVR Studio or the newer Atmel Studio, may also be used to develop software for the Arduino.

Arduino is open-source hardware: the Arduino hardware reference designs are distributed under a Creative Commons Attribution Share-Alike 2.5 license and are available on the Arduino Web site. Layout and production files for some versions of the Arduino hardware are also available. The source code for the IDE is available and released under the GNU General Public License, version 2.

Although the hardware and software designs are freely available under copyleft licenses, the developers have requested that the name "Arduino" be exclusive to the official product and not be used for derivative works without permission. The official policy document on the use of the Arduino name emphasizes that the project is open to incorporating work by others into the official product.[21] Several Arduino-compatible products commercially released have avoided the "Arduino" name by using "-duino" name variants.

c. Arduino Gsm Shield

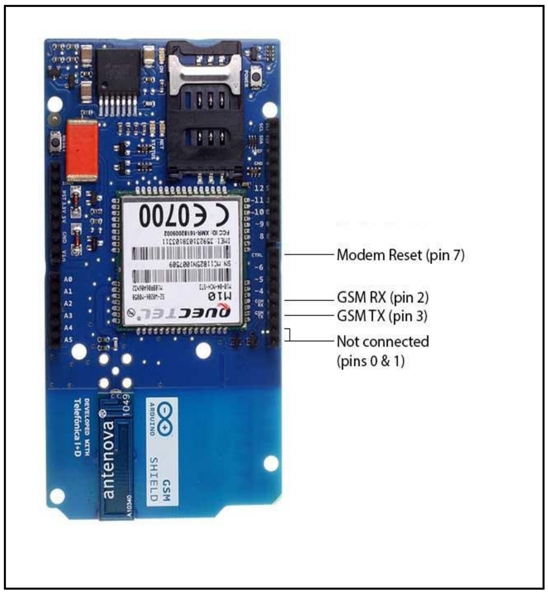
The Arduino GSM Shield connects your Arduino to the internet using the GPRS wireless network. Just plug this module onto your Arduino board, plug in a SIM card from an operator offering GPRS coverage and follow a few simple instructions to start controlling your world through the internet. You can also make/receive voice calls (you will need an external speaker and microphone circuit) and send/receive SMS messages.

The Arduino GSM Shield allows an Arduino board to connect to the internet, make/receive voice calls and send/receive SMS messages. The shield uses a radio modem M10 by Quectel (datasheet). It is possible to communicate with the board using AT commands. The GSM library has a large number of methods for communication with the shield.

The shield uses digital pins 2 and 3 for software serial communication with the M10. Pin 2 is connected to the M10’s TX pin and pin 3 to its RX pin. See these notes for working with an Arduino Mega, Mega ADK, or Leonardo. The modem's PWRKEY pin is connected to Arduino pin 7.

The M10 is a Quad-band GSM/GPRS modem that works at frequencies GSM850MHz, GSM900MHz, DCS1800MHz andPCS1900MHz. It

supports TCP/UDP and HTTP protocols through a GPRS connection. GPRS data downlink and uplink transfer speed maximum is 85.6 kbps.
To interface with the cellular network, the board requires a SIM card provided by a network operator. See the getting started page for additional information on SIM usage.
The most recent revision of the board uses the 1.0 pinout on rev 3 of the Arduino Uno board.



The GSM shield comes bundled with a SIM from Telefonica/Movilforum that will work well for developing machine to machine (M2M) applications. It is not necessary to use this specific card with the shield. You may use any SIM that works on a network in your area.

The Movilforum SIM card includes a roaming plan. It can be used on any supported GSM network. There is coverage throughout the Americas and Europe for this SIM, check the Movilforum service availability page for specific countries that have supported networks.

Activation of the SIM is handled by Movilforum. Detailed instructions on how to register and activate your SIM online and add credit are included on a small pamphlet that comes with your shield. The SIM must be inserted into a powered GSM shield that is mounted on an Arduino for activation.

These SIM card come without a PIN, but it is possible to set one using the GSM library's GSMPIN class.

You cannot use the included SIM to place or receive voice calls.

You can only place and receive SMS with other SIMs on the Movilforum network.

It's not possible to create a server that accepts incoming requests from the public internet. However, the Movilforum SIM will accept incoming requests from other SIM cards on the Movilforum network.

For using the voice, and other functions of the shield, you'll need to find a different network provider and SIM. Operators will have different policies for their SIM cards, check with them directly to determine what types of connections are supported.

Power requirements

It is recommended that the board be powered with an external power supply that can provide between 700mA and 1000mA. Powering an Arduino and the GSM shield from a USB connection is not recommended, as USB cannot provide the required current for when the modem is in heavy use.
The modem can pull up to 2A of current at peak usage, which can occur during data transmission. This current is provided through the large orange capacitor on the board's surface.

On board indicators

The shield contains a number of status LEDs:

- On: shows the Shield gets power.
- Status: turns on to when the modem is powered and data is being transferred to/from the GSM/GPRS network.

- Net: blinks when the modem is communicating with the radio network.

On board interfaces

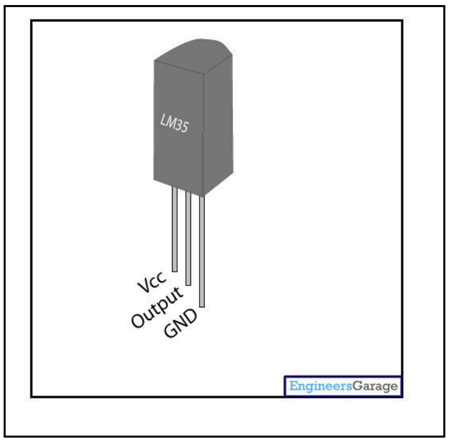
The shield supports AIN1 and AOUT1 as audio interfaces; an analog input channel and an analog output channel. The input, exposed on pins MIC1P/MIC1N, can be used for both microphone and line inputs. An electret microphone can be used for this interface. The output, exposed as lines SPK1P/SPK1N, can be used with either a receiver or speaker. Through the modem, it is possible to make voice calls. In order to speak to and hear the other party, you will need to add a speaker and microphone.

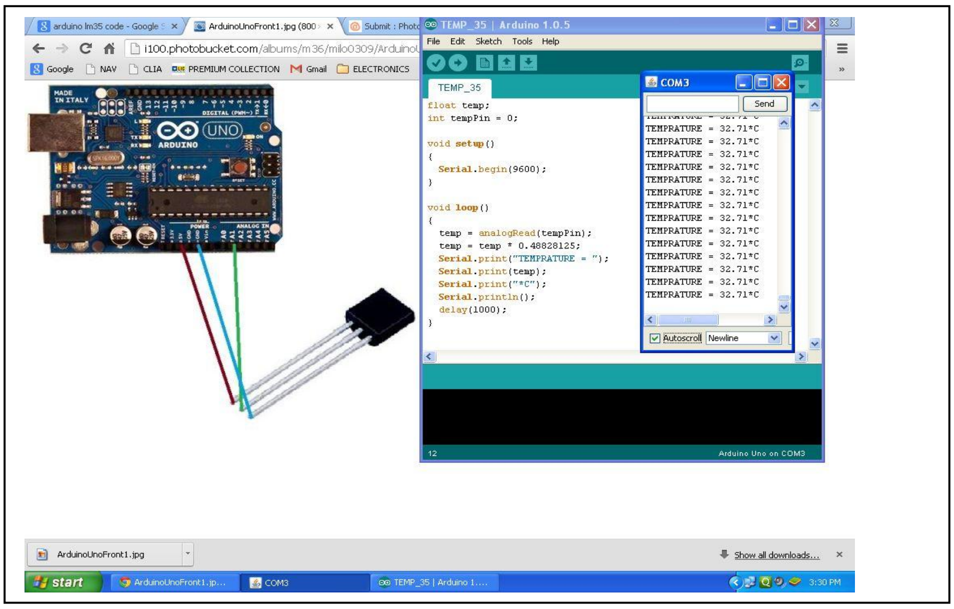
Lcd Shield

The *LCD Keypad shield* is developed for Arduino compatible boards, to provide a user-friendly interface that allows users to go through the menu, make selections etc. It consists of a 1602 white character blue backlight LCD. The keypad consists of 5 keys — select, up, right, down and left. To save the digital IO pins, the keypad interface uses only one ADC channel. The key value is read through a 5 stage voltage divider.

Lm 35 Sensor

LM35 is a precision IC temperature sensor with its output proportional to the temperature (in ^{o}C). The sensor circuitry is sealed and therefore it is not subjected to oxidation and other processes. With **LM35**, temperature can be measured more accurately than with a thermistor. It also possess low self heating and does not cause more than 0.1 ^{o}C temperature rise in still air.



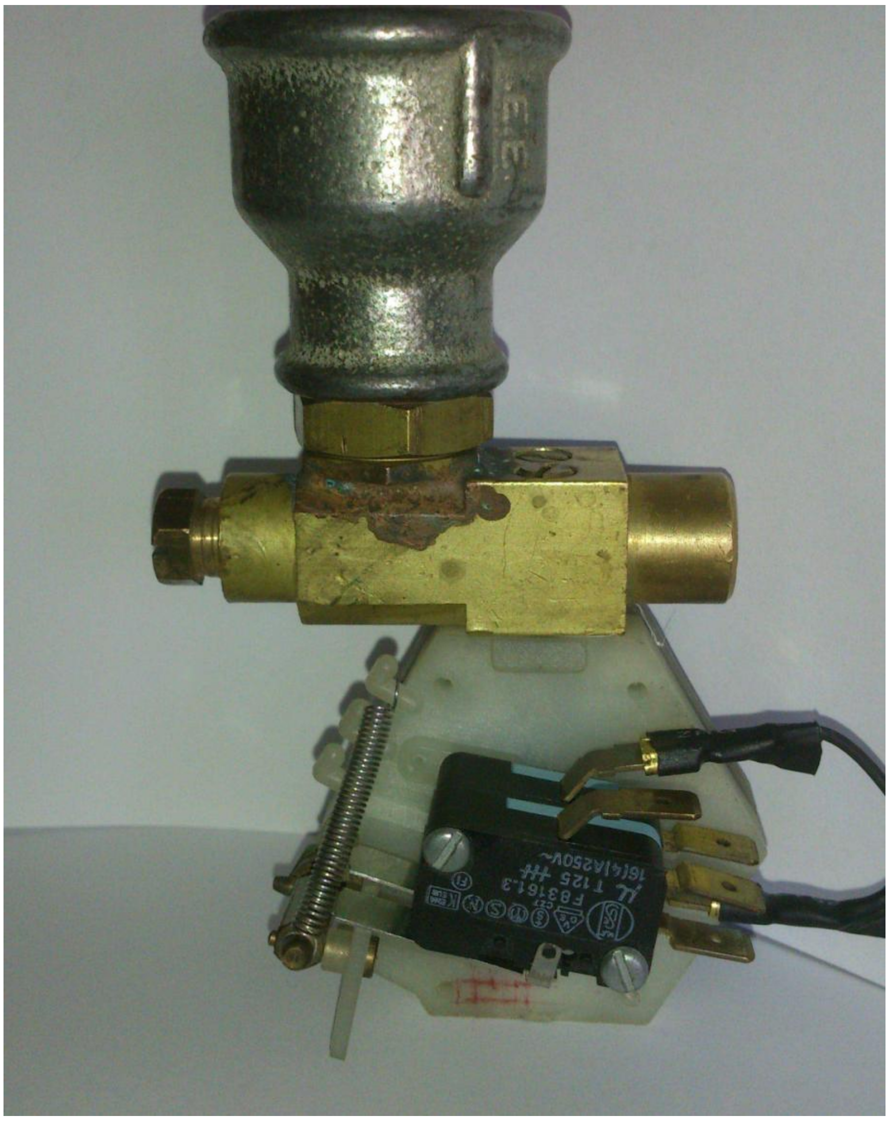


```
float temp;
int tempPin = 0;

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  temp = analogRead(tempPin);
  temp = temp * 0.48828125;
  Serial.print("TEMPRATURE = ");
  Serial.print(temp);
  Serial.print("*C");
  Serial.println();
  delay(1000);
}
```

```
TEMPRATURE = 32.71*C
TEMPRATURE = 32.71*C
TEMPRATURE = 32.71*C
TEMPRATURE = 32.71*C
TEMPRATURE = 32.71*C
TEMPRATURE = 32.71*C
TEMPRATURE = 32.71*C
TEMPRATURE = 32.71*C
TEMPRATURE = 32.71*C
TEMPRATURE = 32.71*C
TEMPRATURE = 32.71*C
TEMPRATURE = 32.71*C
TEMPRATURE = 32.71*C
TEMPRATURE = 32.71*C
```

- F83161.3 switch for water pressure:

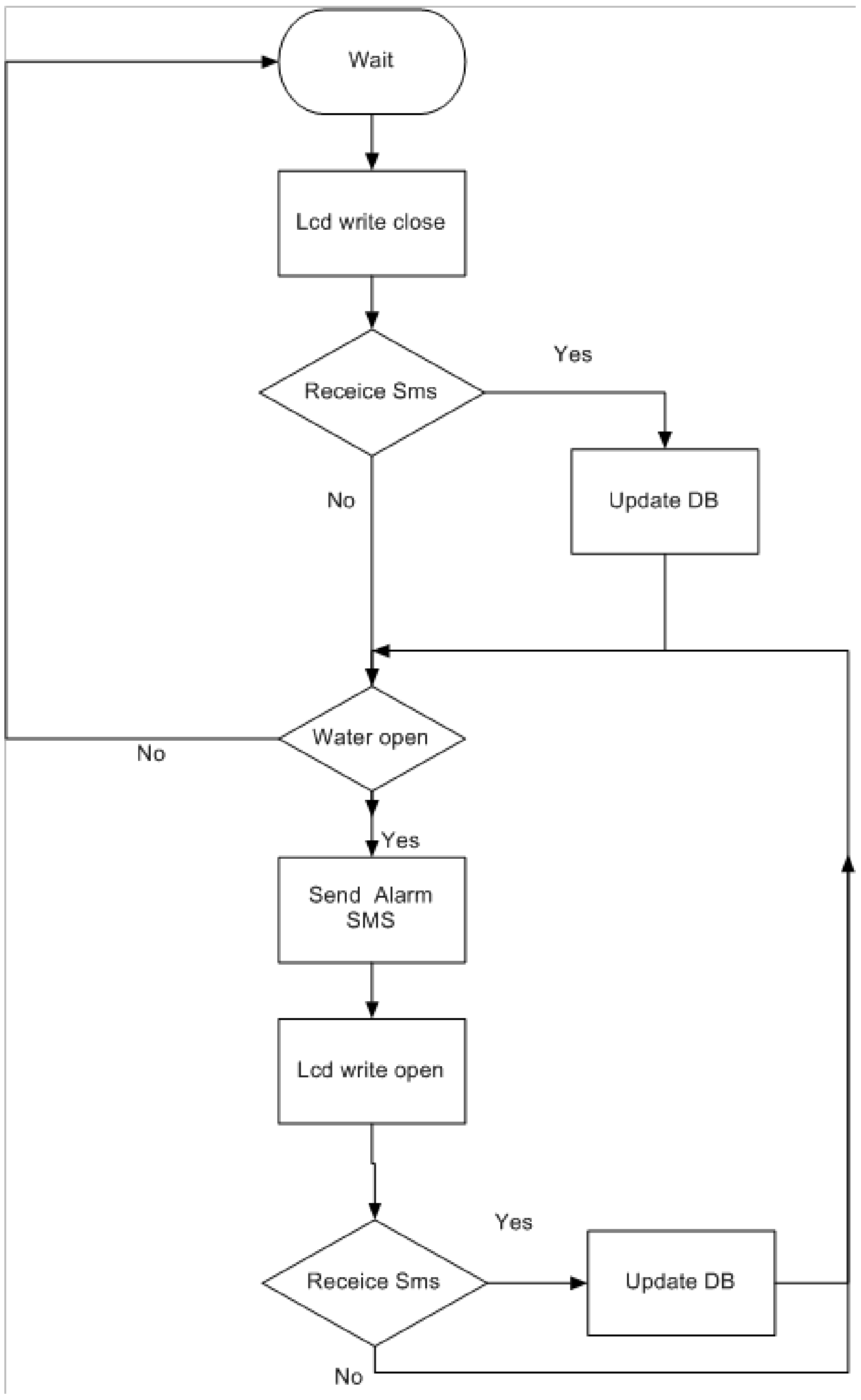
When water pressure is increased switch open



Chapter 3. Code

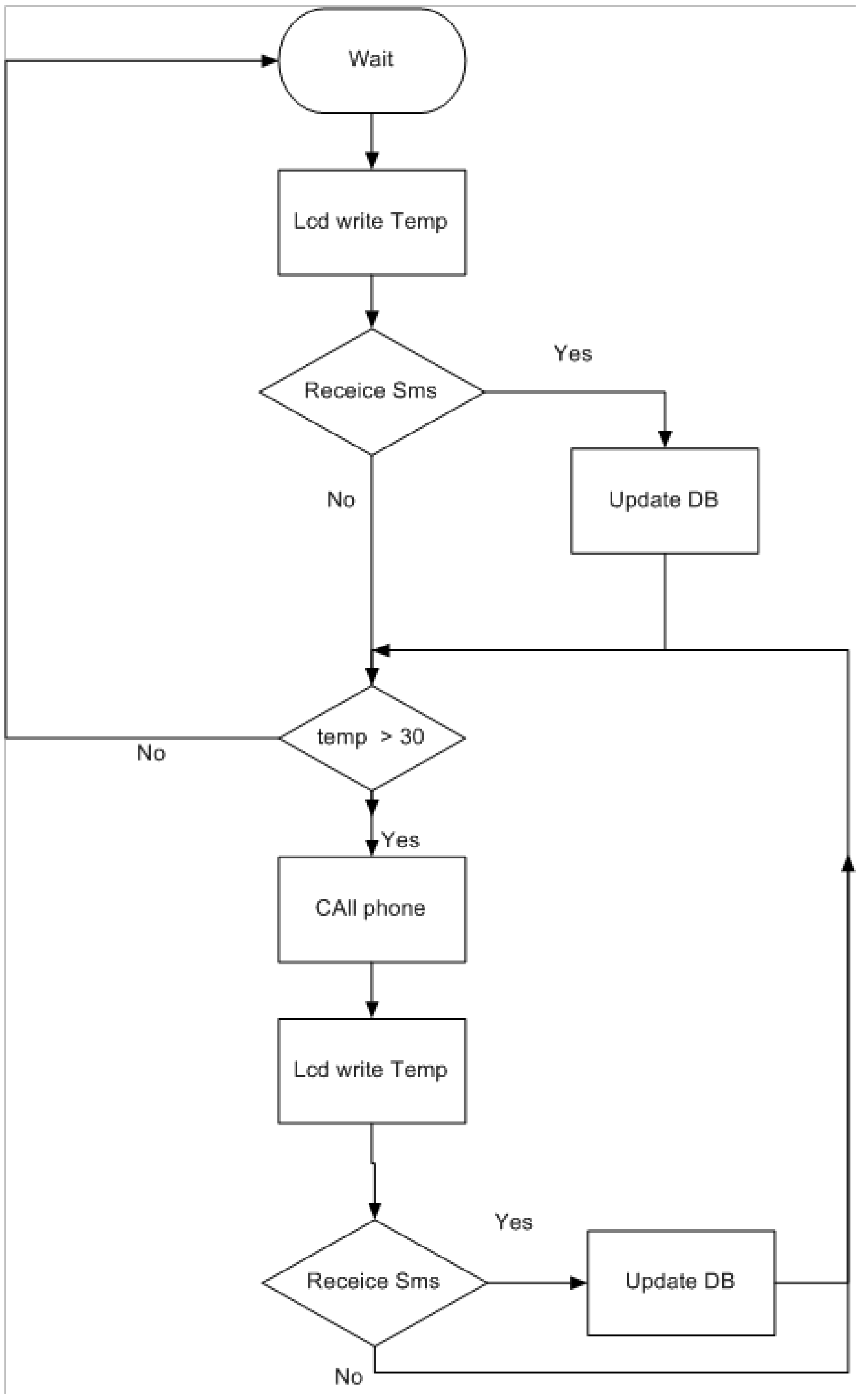
Flow Diagram for Water pressure control

The system is in standby for as long as the water pressure switch is closed, when received any sms then update Database for the longer is the situation of that moment.
When the switch water pressure open up it is sent sms to the user informing him that there is water available.


Wait
Lcd write close
Receice Sms
Yes
No
Update DB
Water open
No
Yes
Send Alarm SMS
Lcd write open
Receice Sms
Yes
Update DB
No

Flow Diagram for Temperature control

The system is in standby as long as the temperature is below 30 degrees Celsius, when received any sms then update the DataBase for the longer is the temperature at that moment.
When the temperature exceeds 30 degrees then calls the user.



AgriArduino code for water pressure

Highlights of the code

Variables to hold the server, path.

```
char server[] = "angelaki.gr";
char path[] = "/arduino/get.php?action=insert&sensor1=";
```

```
int port = 80;
```

Port 80 is the port that the server "listens to" or expects to receive from a Web client, assuming that the default was taken when the server was configured or set up.

The temperature sensor LM35 linked with A1 (analog) pin of Arduino.

```
int tempPin = A1;
```

The Water Pressure Switch linked with A3 (analog) pin of Arduino.

```
int sensorValue = analogRead(A3);
```

Checks for as much water pressure switch is open execute orders that are within the loop.

```
 while (sensorValue > 1010 ){
```

Calls the function through which it will be sent sms.

```
sendsmsFunction();
```

Calls the function through which it will be update DataBase.

```
UpdateDBFunction(sw);
```

Read temperature and to convert degrees Celsius.

```
temp = analogRead(tempPin);
temp = temp/2;
```

Make a voice call.

```
vcs.voiceCall("**********");
```

Important to connect and sent data to the DataBase needs to connect to the specified Access Point Name (APN) to initiate GPRS communication.

Every cellular provider has an Access Point Name (APN) that serves as a bridge between the cellular network and the internet. Sometimes, there is a username and password associated with the connection point. For example, the Bluevia APN is bluevia.movistar.es, but it has no password or login name.

Syntax

grps.attachGPRS(APN, user, password)

Parameters

- APN : char array, the Access Point Name (APN) provided by the mobile operator
- user : char array, the username for the APN
- password : char array, the password to access the APN

```
(gprs.attachGPRS("gint.b-online.gr", "", "")
```

```
// libraries
#include <GSM.h>

#include <LiquidCrystal.h>

// PIN Number
#define PINNUMBER ""

// APN data
#define GPRS_APN       "" // replace your GPRS APN
#define GPRS_LOGIN     ""    // replace with your GPRS login
#define GPRS_PASSWORD  "" // replace with your GPRS password

// initialize the library instance
GSMClient client;
GPRS gprs;
GSM gsmAccess; // include a 'true' parameter for debug enabled

LiquidCrystal lcd(8, 9, 4, 5, 6, 7);
```

```
// initialize the library instance

GSM_SMS sms;
GSM_SMS sms1;
GSMVoiceCall vcs;
int temp;
int temp1;
int tempPin = A1;
int flag=0;

int count;
String sw="open";
char senderNumber[20];

// URL, path & port \
char server[] = "angelaki.gr";
char path[] = "/arduino/get.php?action=insert&sensor1=";
int port = 80; // 80 for HTTP

const int buttonPin = A3;     // the number of the pushbutton pin
const int ledPin =  13;      // the number of the LED pin

// variables will change:
int buttonState = 0;         // variable for reading the pushbutton status
int temp_analogPin = A1;
char remoteNumber[20];
void setup()
{

  Serial.begin(9600);
   while (!Serial) {
                   ;
            }
  Serial.println("SMS Messages Receiver");

  // connection state

}

void loop()
{
   char c;
   int sensorValue = analogRead(A3);
   buttonState = analogRead(buttonPin);

  Serial.println("test ");



    lcd.clear();
    lcd.setCursor(0, 1);
    lcd.print("water:"  );



    if (sensorValue > 1010) {
        lcd.clear();
        lcd.setCursor(0, 1);
        lcd.print("water:"  );
        lcd.setCursor(8, 1);
        sw = "open";
        lcd.print("open");
        Serial.print("open ");

    }
    else {
        lcd.clear();
        lcd.setCursor(0, 1);
        lcd.print("water:"  );
        lcd.setCursor(8, 1);
        sw = "close";
        lcd.print("close");
   }



    delay(1000);

   boolean notConnected = true;

// Start GSM connection
while (notConnected)
{
         if (gsmAccess.begin(PINNUMBER) == GSM_READY)
         notConnected = false;
         else
   {
  Serial.println("Not connected");
  delay(1000);
   }
}

    Serial.println("Sensor Water: ");
    Serial.println("GSM initialized");
    Serial.println("Waiting for messages");

  while (sms.available())
{
      Serial.println("Message received from:");

   // Get remote number
      sms.remoteNumber(senderNumber, 20);
```

```
        Serial.println(senderNumber);

    // An example of message disposal
    // Any messages starting with # should be discarded
          if (sms.peek() == '#')
          {
            Serial.println("Discarded SMS");
            sms.flush();
          }

    // Read message bytes and print them

    while (c = sms.read())

      Serial.print(c);
      Serial.println("\nEND OF MESSAGE");
      sms.flush();
      Serial.println("db");
      UpdateDBFunction(sw);
      Serial.println("MESSAGE DELETED");
  }

   //  Serial.println(sensorValue);

     while (sensorValue > 1010 ){

           lcd.clear();
           lcd.setCursor(0, 1);
           lcd.print("water:"  );
           lcd.setCursor(8, 1);
           sw = "open";
           lcd.print("open");
           Serial.println("sms");
    sendsmsFunction();
    vcs.hangCall();
    Serial.println("hang");
    lcd.setCursor(10, 0);
    if (sms.available())
     {
       Serial.println("Message received from:");

// Get remote number
       sms.remoteNumber(senderNumber, 20);
       Serial.println(senderNumber);

// An example of message disposal
// Any messages starting with # should be discarded
       if (sms.peek() == '#')
       {
         Serial.println("Discarded SMS");
         sms.flush();
        }


     while (c = sms.read())

       Serial.print(c);

       Serial.println("\nEND OF MESSAGE");

       sms.flush();
       UpdateDBFunction(sw);
       Serial.println("MESSAGE DELETED");
   }
```

```
          delay(1000);
          sensorValue = analogRead(A3);
          if (sensorValue > 1010) {
              lcd.setCursor(8, 1);
              sw = "open";
              lcd.print("open");
              Serial.print("open ");

          }
          else {
              lcd.setCursor(8, 1);
              sw = "close";
              lcd.print("close");

            }
setup();
         }

}

int readSerial(char result[])
{
  int i = 0;
  while(1)
  {
    while (Serial.available() > 0)
    {
      char inChar = Serial.read();
      if (inChar == '\n')
     {
        result[i] = '\0';
        Serial.flush();
        return 0;
      }
      if(inChar!='\r')
      {
        result[i] = inChar;
        i++;
      }
    }
  }
}
```

AgriArduino code for Temperature

```

// libraries
#include <GSM.h>

#include <LiquidCrystal.h>

// PIN Number
#define PINNUMBER ""

// APN data
#define GPRS_APN       "" // replace your GPRS APN
#define GPRS_LOGIN     ""    // replace with your GPRS login
#define GPRS_PASSWORD  "" // replace with your GPRS password

// initialize the library instance
GSMClient client;
GPRS gprs;
GSM gsmAccess; // include a 'true' parameter for debug enabled

LiquidCrystal lcd(8, 9, 4, 5, 6, 7);
// initialize the library instance
```

```
GSM_SMS sms;
GSMVoiceCall vcs;
int temp;
int temp1;
int tempPin = A1;
int flag=0;

int count;
int st;
String sw="open";
char senderNumber[20];

// URL, path & port \
char server[] = "angelaki.gr";
char path[] = "/arduino/get.php?action=insert&sensor1=";
int port = 80; // 80 for HTTP

//const int buttonPin = A3;     // the number of the pushbutton pin
//const int ledPin =  13;      // the number of the LED pin

// variables will change:
int buttonState = 0;         // variable for reading the pushbutton status
int temp_analogPin = A1;
char remoteNumber[20];
void setup()
{

  Serial.begin(9600);
        while (!Serial) {
                ;
         }
  Serial.println("SMS Messages Receiver");
  // connection state
  boolean notConnected = true;

  // Start GSM connection
  while (notConnected)
  {
    if (gsmAccess.begin(PINNUMBER) == GSM_READY)
      notConnected = false;
    else
    {
		Serial.println("Not connected");
		delay(1000);
    }
  }

  Serial.println("GSM initialized");

}



void loop()
{

  char c;
    temp = analogRead(tempPin);

   temp = temp/2;

     lcd.clear();
     lcd.setCursor(0, 1);
     lcd.print("Celsius:"  );
   lcd.setCursor(10, 0);
   lcd.print(temp);
   st = temp;

     temp = analogRead(tempPin);
    temp = temp/2;

    Serial.println(temp);
    delay(1000);

 boolean notConnected = true;

// Start GSM connection
while (notConnected)
 {
  if (gsmAccess.begin(PINNUMBER) == GSM_READY)
    notConnected = false;
  else
  {
          Serial.println("Not connected");
          delay(1000);
  }
}

   Serial.println("Sensor Water: ");
   Serial.println("GSM initialized");
   Serial.println("Waiting for messages");

 while (sms.available())
{
      Serial.println("Message received from:");
  // Get remote number
      sms.remoteNumber(senderNumber, 20);
      Serial.println(senderNumber);

  // An example of message disposal
  // Any messages starting with # should be discarded
        if (sms.peek() == '#')
        {
          Serial.println("Discarded SMS");
          sms.flush();
        }

  // Read message bytes and print them

  while (c = sms.read())

    Serial.print(c);
    Serial.println("\nEND OF MESSAGE");
    sms.flush();
    Serial.println("db");
    UpdateDBFunction();
    Serial.println("MESSAGE DELETED");



 //  Serial.println(sensorValue);

    while (temp > 30 ){
      temp = analogRead(tempPin);
      temp = temp/2;
      Serial.println(temp);
      delay(1000);
      if (temp > 30){
          vcs.voiceCall("**********");
          Serial.println("call");



          vcs.hangCall();

          lcd.setCursor(10, 0);






         }
  setup();
        }

      }
}
```

UpdateDBFunction

```
void UpdateDBFunction(int temp,String watersw)
{
   boolean notConnected = true;

  // Start GSM shield
  // If your SIM has PIN, pass it as a parameter of begin() in quotes
  while(notConnected)
  {
    if((gsmAccess.begin(PINNUMBER)==GSM_READY) &
        (gprs.attachGPRS("gint.b-online.gr", "", "")==GPRS_READY))
      notConnected = false;
    else
    {
      Serial.println("Not connected");
      delay(1000);
    }
  }

  Serial.println("connecting...");

  // if you get a connection, report back via serial:
  if (client.connect(server, port))
  {
    Serial.println("connected");
    // Make a HTTP request:
    client.print("GET http://angelaki.gr");
    client.print(path);
    client.print(temp);
    client.print("&sensor2=");
    client.print(watersw);
    client.println(" HTTP/1.0");
    client.println();
  }
  else
  {
    // if you didn't get a connection to the server:
    Serial.println("connection failed");
  }

}
```

sendsmsFunction

```
void sendsmsFunction(String  txtMsg)
{
   boolean notConnected = true;

int str_len = txtMsg.length() + 1;

// Prepare the character array (the buffer)
char char_array[str_len];

// Copy it over
txtMsg.toCharArray(char_array, str_len);

  // Start GSM shield
  // If your SIM has PIN, pass it as a parameter of begin() in quotes
  while(notConnected)
  {
    if(gsmAccess.begin(PINNUMBER)==GSM_READY)
      notConnected = false;
    else
    {
      Serial.println("Not connected");
      delay(1000);
    }
  }
  Serial.println("GSM initialized");

  sms.beginSMS(remoteNum);
  sms.print(char_array);
  sms.endSMS();
  Serial.println("\nCOMPLETE!\n");
   return;
}
```

Chapter 4: AgriArduino Update Database

This project which is based on arduino first of all collect data. These data should initially be able to store and classified to derive results.

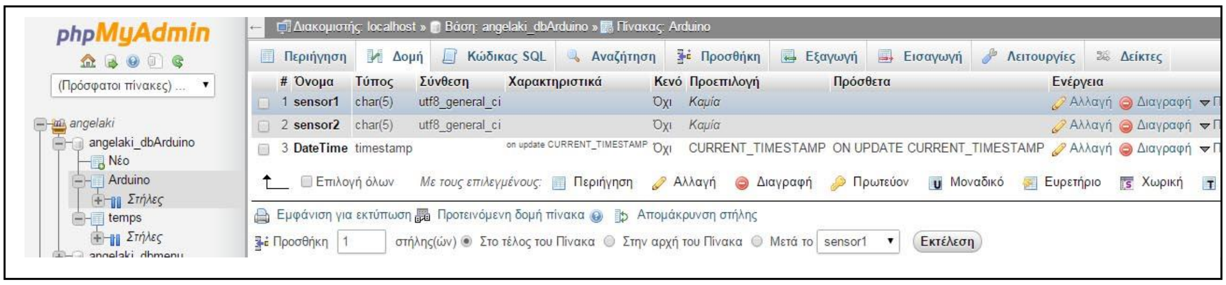
AgriArduino stores the data from sensors in WebDatabase. The database is located on a server with a domain name www.angelaki.gr.

Listed below are the steps taken to implement the storage of data from the database arduino.

Create a Database with Mysql

Mysql contained in the server "tools". Create easy database with name "angelaki_dbArduino" and name of user "angelaki_user" , password:12345.

In the phpMyAdmin Create a Table with sql or tools from phpMyAdmin panel.



Διακομιστής: localhost » Βάση: angelaki_dbArduino » Πίνακας: Arduino

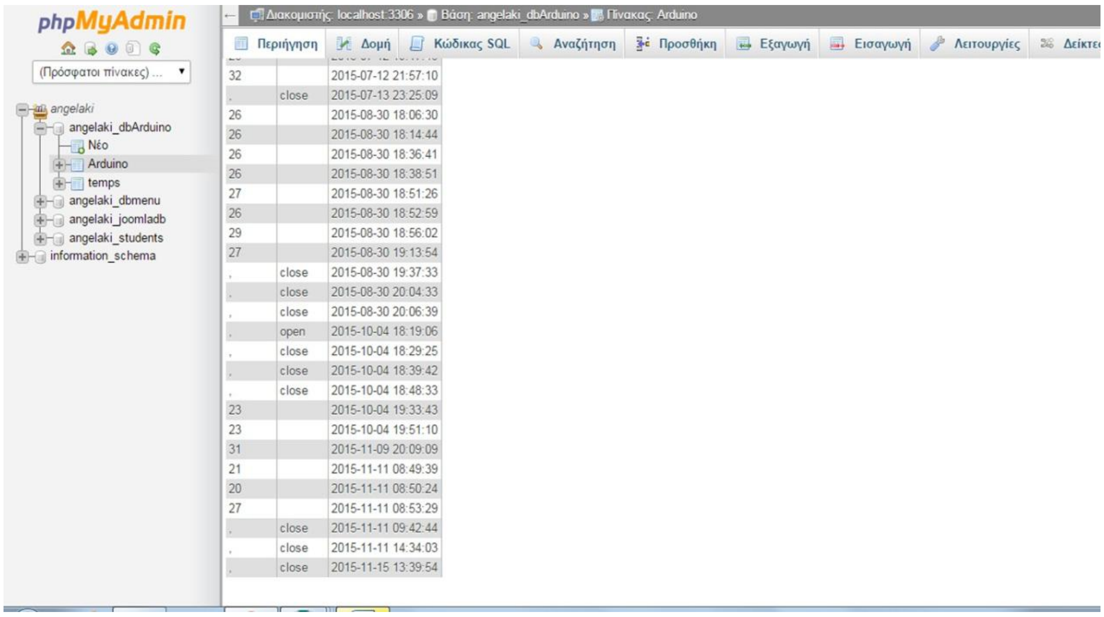
	#	Όνομα	Τύπος	Σύνθεση	Χαρακτηριστικά	Κενό	Προεπιλογή	Πρόσθετα	Ενέργεια
☐	1	**sensor1**	char(5)	utf8_general_ci		Όχι	*Καμία*		Αλλαγή Διαγραφή
☐	2	**sensor2**	char(5)	utf8_general_ci		Όχι	*Καμία*		Αλλαγή Διαγραφή
☐	3	**DateTime**	timestamp		on update CURRENT_TIMESTAMP	Όχι	CURRENT_TIMESTAMP	ON UPDATE CURRENT_TIMESTAMP	Αλλαγή Διαγραφή

☐ Επιλογή όλων *Με τους επιλεγμένους:* Περιήγηση Αλλαγή Διαγραφή Πρωτεύον Μοναδικό Ευρετήριο Χωρική

Εμφάνιση για εκτύπωση Προτεινόμενη δομή πίνακα Απομάκρυνση στήλης

Προσθήκη 1 στήλης(ών) [✓ Στο τέλος του Πίνακα] Στην αρχή του Πίνακα Μετά το sensor1 Εκτέλεση

In this table we need a field for each sensor and a field that stores the date and time are updated.



32		2015-07-12 21:57:10
.	close	2015-07-13 23:25:09
26		2015-08-30 18:06:30
26		2015-08-30 18:14:44
26		2015-08-30 18:36:41
26		2015-08-30 18:38:51
27		2015-08-30 18:51:26
26		2015-08-30 18:52:59
29		2015-08-30 18:56:02
27		2015-08-30 19:13:54
,	close	2015-08-30 19:37:33
.	close	2015-08-30 20:04:33
,	close	2015-08-30 20:06:39
.	open	2015-10-04 18:19:06
,	close	2015-10-04 18:29:25
.	close	2015-10-04 18:39:42
,	close	2015-10-04 18:48:33
23		2015-10-04 19:33:43
23		2015-10-04 19:51:10
31		2015-11-09 20:09:09
21		2015-11-11 08:49:39
20		2015-11-11 08:50:24
27		2015-11-11 08:53:29
.	close	2015-11-11 09:42:44
,	close	2015-11-11 14:34:03
.	close	2015-11-15 13:39:54

Create a php file to connect and insert data from AgriArduino

First we connect with database with “conect.php” file

```
<?php
// Create connection
$con=mysqli_connect("angelaki.gr","angelaki_user","12345");

// Check connection
if (mysqli_connect_errno()) {
  echo "Failed to connect to MySQL: " . mysqli_connect_error();
}

else echo "ok"
?>
```

File “get.php” store the data in the DataBase.

```
<?php

$sensor1= $_GET['sensor1'];

echo "sensor1: ".$sensor1;

$sensor2= $_GET['sensor2'];

echo "sensor2: ".$sensor2;
$now = time();
$num = date("w");
if ($num == 0)
{ $sub = 6; }
else { $sub = ($num-1); }
$WeekMon   =  mktime(0,  0,  0,  date("m",  $now)    ,  date("d",  $now)-$sub,
date("Y", $now));    //monday week begin calculation
$todayh = getdate($WeekMon); //monday week begin reconvert

//$d = $todayh[mday];
//$m = $todayh[mon];
//$y = $todayh[year];
//echo "$d-$m-$y"; //getdate converted day

$d=strtotime("today");
echo date("Y-m-d", $d) . "<br>";

            /*
        echo "<pre>";
        var_dump($_REQUEST);
           echo "</pre>";
        */
           $action = null;
           if (array_key_exists("action", $_REQUEST)){
             $action = $_REQUEST["action"];          }
```

```
      //1. CONNECT TO DATABASE
      $db_server = "angelaki.gr";
      $db_database = "angelaki_dbArduino";
      $db_username = "angelaki_user";
      $db_password = "12345";


      mysql_connect($db_server,$db_username,$db_password);
      @mysql_select_db($db_database) or die( "Unable to select database");

      mysql_query("set names utf8");

      if ($action == "insert"){

        $sensor1= $_REQUEST["sensor1"];
      $sensor2= $_REQUEST["sensor2"];


      $query = "INSERT INTO Arduino (sensor1,sensor2) VALUES
('{$sensor1}', '{$sensor2}')";

      mysql_query($query);
   }else if ($action == "delete"){
      $sensor1= $_REQUEST["sensor1"];
```

```
        $query = "DELETE FROM dbMenu WHERE sensor1= {$sensor1}";
        mysql_query($query);
    } // end if



$blugh = "\n";
$k = ",";

$myFile = "data.txt";//save data in the text file for chart.

$fh = fopen($myFile, 'a') or die("can't open file");

fwrite($fh, $blugh);
fwrite($fh, date("Y-m-d", $d));
fwrite($fh, $k);
fwrite($fh, $sensor1);

//fwrite($fh, $sensor2);

fclose($fh);
?>
```

File “menu.php” display the data from angelaki_dbArduino in a browser.

```
<html>
          <head>
                    <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8" />
          </head>
          <body background="http://angelaki.gr/test/images/body-
background.gif" >
    <h1>Menu:</h1>
<?php
```

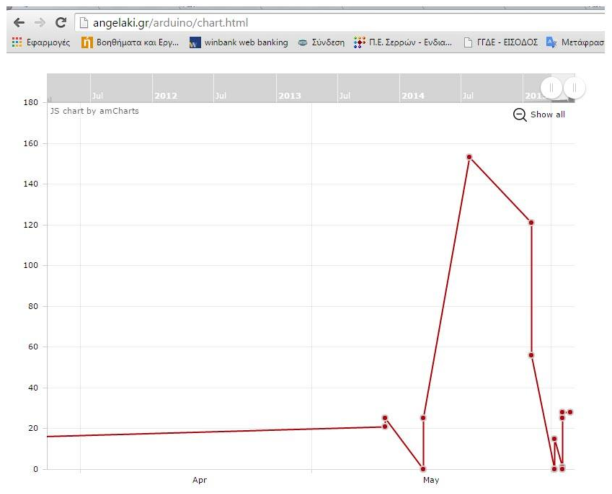
```
          /*
       echo "<pre>";
       var_dump($_REQUEST);
         echo "</pre>";
       */

         $action = null;
         if (array_key_exists("action", $_REQUEST)){
           $action = $_REQUEST["action"];
         }
       //1. CONNECT TO DATABASE
       $db_server = "angelaki.gr";
       $db_database = "angelaki_dbArduino";
       $db_username = "angelaki_user";
       $db_password = "12345";

       mysql_connect($db_server,$db_username,$db_password);
       @mysql_select_db($db_database) or die( "Unable to select database");

       mysql_query("set names utf8");

       if ($action == "insert"){
         $sensor1= $_REQUEST["sensor1"];
       $sensor2= $_REQUEST["sensor2"];
       $DateTime= $_REQUEST["DateTime"];

       $query = "INSERT INTO Arduino (sensor1, sensor2, DateTime) VALUES
('{$sensor1}', '{$sensor2}', '{$DateTime}')";

       mysql_query($query);
    }else if ($action == "delete"){
       $DateTime = $_REQUEST["DateTime"];
       $query = "DELETE FROM Arduino WHERE DateTime = {$DateTime}";
```

```
        mysql_query($query);
    } // end if
                    $query = "SELECT * FROM Arduino";
          $result = mysql_query($query);
    $count = 0;
    while ($row = mysql_fetch_assoc($result)){
                    //var_dump($row);
                    $count++;

                      $DateTime= $row["DateTime"];

          $sensor1 = $row["sensor1"];
          $sensor2 = $row["sensor2"];


                    echo        "<br/>\n              <b>DateTime.{$DateTime}<br/>\n
<b>Temperature.</b>:{$sensor1}<br/>\n                         <b>Sensor2.</b>:
{$sensor2}<br/>\n";

?>
          <br/><br/><br/>
<?php

} // end while
?>
</html>
```

a. <u>Online Chart with results.</u>

Results display in the user with file “chart.html”.

This read the data.txt file and shows in graphical form.



<u>Arduino Code to send data from sensors</u>

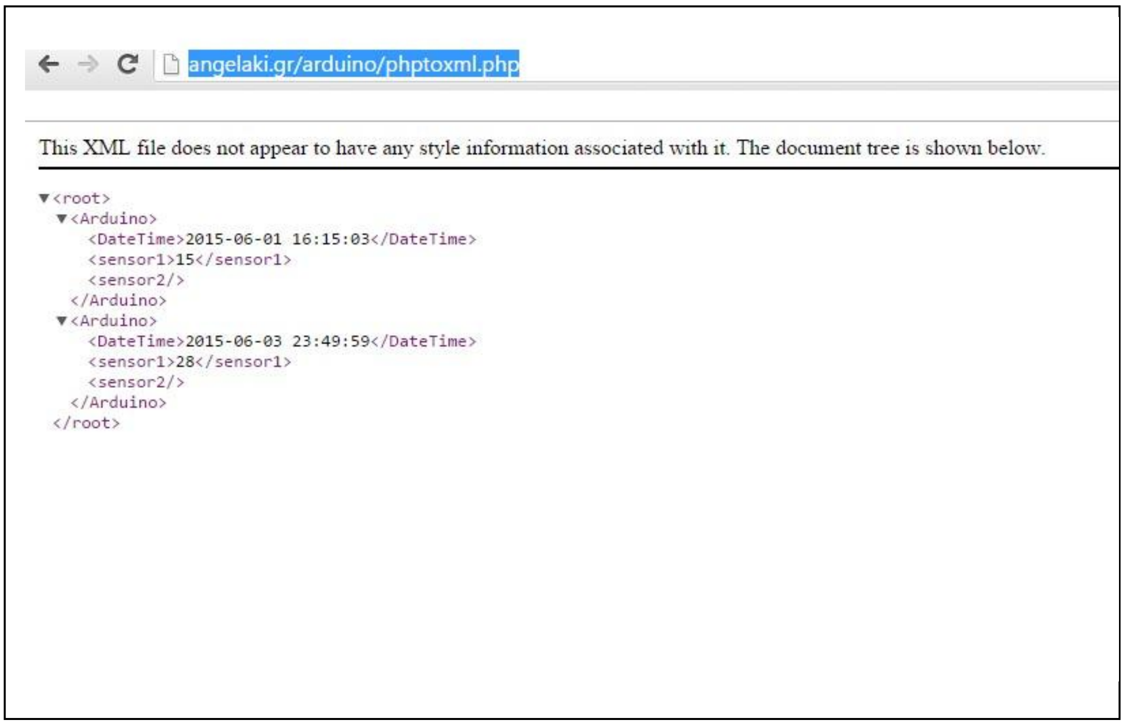
AgriArduino send the data for temperature with call the function “*UpdateDBFunction*”.

```
void UpdateDBFunction(int temp)
{
   boolean notConnected = true;

  // Start GSM shield
  // If your SIM has PIN, pass it as a parameter of begin() in quotes
  while(notConnected)
  {
    if((gsmAccess.begin(PINNUMBER)==GSM_READY) &
        (gprs.attachGPRS("gint.b-online.gr", "", "")==GPRS_READY))
      notConnected = false;
    else
  {
    Serial.println("Not connected");
    delay(1000);
  }
}
Serial.println("connecting...");
// if you get a connection, report back via serial:
if (client.connect(server, port))
{
  Serial.println("connected");
  // Make a HTTP request:
  client.print("GET http://angelaki.gr");
  client.print(path);
  client.print(temp);

  client.println(" HTTP/1.0");
  client.println();
}
else
{
  // if you didn't get a connection to the server:
  Serial.println("connection failed");
}
   }
```

Chapter 5:Android Application AgriArduino

Agri Arduino is a application which can display the data from the database in every smartphones features Android software.
This app developed with Eclipse.

Android app read http://angelaki.gr/arduino/phptoxml.php and show the results.



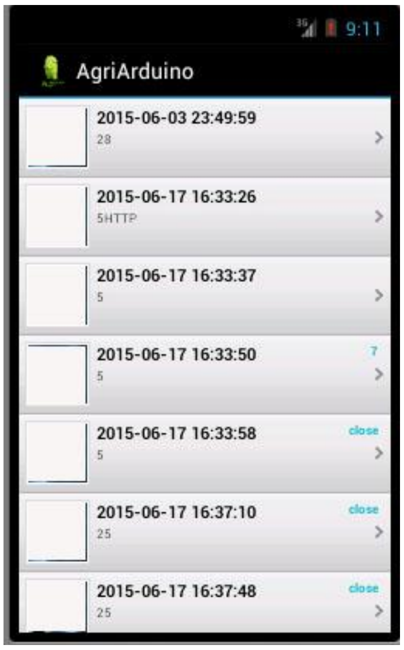
This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<root>
  ▼<Arduino>
     <DateTime>2015-06-01 16:15:03</DateTime>
     <sensor1>15</sensor1>
     <sensor2/>
   </Arduino>
  ▼<Arduino>
     <DateTime>2015-06-03 23:49:59</DateTime>
     <sensor1>28</sensor1>
     <sensor2/>
   </Arduino>
 </root>
```

Phptoxml.php code:

```
<?php
$hostname_conn = "angelaki.gr";
$database_conn = "angelaki_dbArduino";
$username_conn = "angelaki_user";
$password_conn = "12345";
$conn = mysql_pconnect($hostname_conn, $username_conn,
$password_conn) or trigger_error(mysql_error(),E_USER_ERROR);
?>
<?php
// Query the database and get all the records from the Images table
mysql_select_db($database_conn, $conn);
$query_rsImages = "SELECT * FROM Arduino";
$rsImages = mysql_query($query_rsImages, $conn) or die(mysql_error());
$row_rsImages = mysql_fetch_assoc($rsImages);
$totalRows_rsImages = mysql_num_rows($rsImages);

// Send the headers
header('Content-type: text/xml');
header('Pragma: public');
header('Cache-control: private');
header('Expires: -1');
?><?php echo('<?xml version="1.0" encoding="utf-8"?>'); ?>
<root>
  <?php if ($totalRows_rsImages > 0) { // Show if recordset not empty ?>
  <?php do { ?>
          <Arduino>
                    <DateTime><?php echo $row_rsImages['DateTime'];
?></DateTime>
                    <sensor1><?php echo $row_rsImages['sensor1']; ?></sensor1>
                    <sensor2><?php echo $row_rsImages['sensor2']; ?></sensor2>
```

```
        </Arduino>
    <?php } while ($row_rsImages = mysql_fetch_assoc($rsImages)); ?>
        <?php } // Show if recordset not empty ?>
</root>
<?php
mysql_free_result($rsImages);
?>
```



2015-06-03 23:49:59
28

2015-06-17 16:33:26
5HTTP

2015-06-17 16:33:37
5

2015-06-17 16:33:50 7
5

2015-06-17 16:33:58 close
5

2015-06-17 16:37:10 close
25

2015-06-17 16:37:48 close
25

Chapter 6: Scalability and improving

This project has great expandability, things that could be added:

- It could be a solar panel with possibility of charging the battery.
- Sensor detect rain (Water drop Sensor)
- Soil moisture detection sensor
- Arduino gps shield for mode and record the move and safety from theft.
- Cam for image recording.
- Connection solenoid valves for handling catering water from the mobile.

Chapter 7 : Conclusions

The proper use of integrated circuits will have to be a key tool in farmers threatened by climate change, the environment.
The benefits of using integrated circuits for automation, faster update on what is happening in the field.

- reduce production costs
- reduction of working hours
- avoid catastrophic risks
- improve product quality
- enhancing competitiveness
- avoiding the use of pesticides
- outlook and improved rural work

Chapter 8: Bibliography

1. http://en.wikipedia.org/wiki/Arduino *Arduino Uno*

2. http://www.electronique-mixte.fr/kits-de-developpement-processeurs-et-microcontroleurs/kit-arduino/ *Arduino Uno*

3. http://www.arduino.cc/en/Main/ArduinoGSMShield *Gsm Shield*

4. http://www.dfrobot.com/wiki/index.php?title=Arduino_LCD_KeyPad_Shield_(SKU:_DFR0009)#Pin_Allocation *Lcd Shield*

5. http://www.amcharts.com/tutorials/loading-external-data Online chart

6. http://www.w3schools.com/

7. https://www.ictinagriculture.org/sourcebook/module-1-introduction-ict-agricultural-development

8. http://citeseerx.ist.psu.edu