

Electronics playground

Visitekaarduino^{Ver. 1.11}

An Arduino on a business card

Filip Pynckels
Director-general ICT
Federal Public Service Home Affairs
Directorate ICT

December 26, 2019
Update January 26, 2020
Update February 7, 2020

All information in this document is kept under
the MIT open software / hardware license.

Documents:

http://users.telenet.be/pynckels/2019-4-VisitekaArduino_ver_1-1.pdf

Table of contents

Table of contents	2
List of figures	2
List of tables	2
Introduction	3
1. Schematics.....	4
2. Printed Circuit Board (PCB)	5
3. Bill of materials	6
4. Programming the ATmega328	7
4.1. Hardware.....	7
4.2. Software	9
Conclusion.....	11
Appendix 1 - Arduino IDE - Source	12
Appendix 2 - Atmel Studio - Source	13
Appendix 3 – A fire-smoke-temperature warning system	14

List of figures

Figure 1 – Populated Visitekaarduino	3
Figure 2 - Schematics	4
Figure 3 - Printed Circuit Board (PCB).....	5
Figure 4 - ATmega328 pinout.....	7
Figure 5 - ISP interface	7
Figure 6 – Arduino ISP programmer.....	8
Figure 7 - Programming an ATmega328 with an Arduino UNO.....	8
Figure 8 - Arduino IDE - Blink a LED program.....	9
Figure 9 - Atmel Studio - Blink a LED program	10
Figure 10 - Finished Visitekaarduino.....	11
Figure 11 - Fire-smoke-temperature alarm (poc).....	14

List of tables

Table 1 - Visitekaarduino bill of materials.....	6
--	---

Introduction

If you are reading this document, there is a good chance that you have received my [business card](#), have scanned the [QR code](#), and followed the link. Well done!

This document elaborates on the design of my [Visitekaarduino](#) (Dutch for [business card Arduino](#) or something alike). *Figure 1* shows a picture of what a populated card looks like. However, to challenge you, you've only received the blank business card. Assembling it is a challenge I leave to you. This card is powerful enough to serve as the core of a large number of IoT (Internet of Things) and other projects.

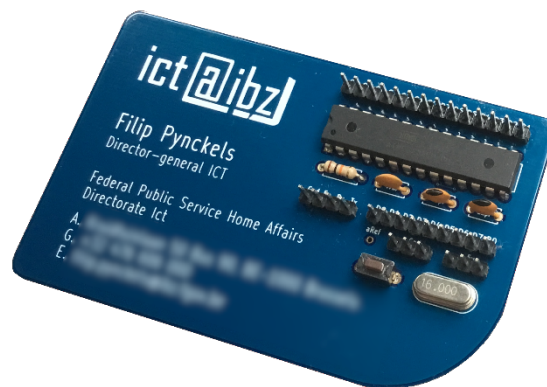


FIGURE 1 – POPULATED VISITEKAARDUINO

Below, all necessary information is given to make it possible to understand the assembling, the programming and the functioning of this Visitekaarduino. This includes the [schematics](#), the [PCB design](#), the [bill of materials](#), some [programming code](#), etc.

The major challenge was to create a price-competitive business card (for volumes from 250 up) that gets more attention than a plain vanilla card. So, the overhead cost, like shipping cost, engineering cost, etc. can be a bit higher if the price per board is low enough. The higher the volumes, the more beneficial this Visitekaarduino becomes.

To reassure you, this card, although small of size, contains the possibility to assemble a full-fledged Arduino that can be used in many [electronics projects](#) or [Internet of Things projects](#). It can also be used to take the first steps in [programming C](#) or C-alike code in the [Arduino IDE](#) programming environment, but also in a professional development environment like [Atmel Studio](#).

As a [proof of concept](#) and coming from the department that does everything that is possible to guarantee the security of the Belgian citizens, a [fire/smoke/temperature alarm](#) is developed with the Visitekaarduino. This proof of concept is extendible to englobe CO2 monitoring, water level monitoring, etc.

I hope that this card provides you with all the fun that meeting me didn't 😊

Figure 2 shows the [schematics of the Visitekaarduino](#). Please note that some security provisions are omitted such as an inductor and reverse biased diode on the AREF port and a capacitor on the reset port. This does not make this Arduino version less flexible; it just requires you not to feed it voltage peaks or powering it with fluctuating DC.

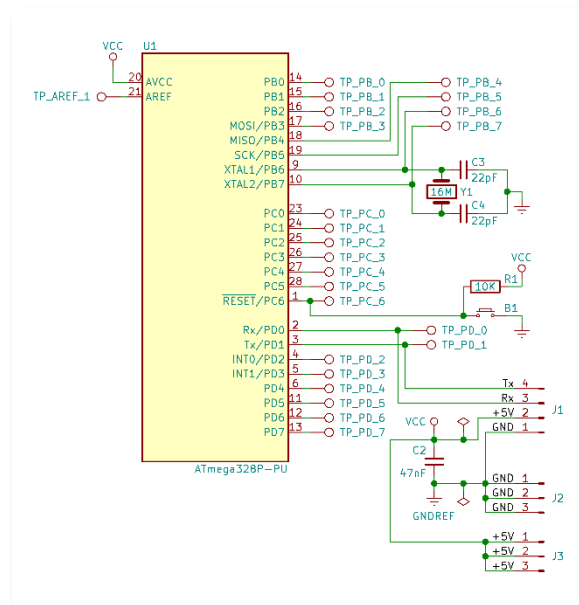


FIGURE 2 - SCHEMATICS

The used **processor is an ATmega328P**, which is about the most used Atmel processor these days. To give our Visitekaarduino a bit of juice, there is place on the PCB to add a 16Mhz crystal with two accompanying capacitors. **You can opt not to add this crystal** and to use the ports B6 and B7 for other goals. Such a configuration limits the processor to 4Mhz clock speed.

Pin C6 is pulled up to Vcc to prevent the processor from going into reset mode due to a floating pin. To permit you to reset the processor, a [reset button](#) can be added. Pulling pin C6 to Gnd resets the processor.

4 of the 6 FTDI pins are broken out, so to permit this little Arduino to communicate with the outside world (Rx on port D0 and Tx on port D1), and to get its power through a variety of power supplies (+5V and Gnd). One option is an USB to TTL cable, another option is a simple 5V transformer. The configuration as given in figure 2 uses less than 100mA. However, if you decide to connect some peripheric devices, this can go up to 500mA (before you get magic smoke). So, caution how much current you source or sink per ATmega328 pin and in total (20mA per pin, 200mA in total) is more important than a powerful transformer. If you wish to source or sink more, some plain vanilla transistors such as the 2N3904 (NPN) or the 2N3906 (PNP) can be of use.

Finally, we have broken out the Vcc (figure 2, J2) and the Gnd (figure 2, J3) power levels to facilitate the connection of the Visitekaarduino to external electronic components.

2. Printed Circuit Board (PCB)

The **printed circuit board design** is shown in *figure 3*. On the left, the frontside of the board is shown, on the right one finds the backside of the board.

The size of the **PCB is conforming to the CR80 standard**: 85.60mm × 53.98mm with a corner radius of 3.175mm. Note that the measurements in *figure 3* are inclusive the edge cut area. The usual thickness of a PCB is 1.6mm but we will opt for a card thickness of 0.4mm to make it more manageable as a business card. The sturdiness of the board will be maintained mostly by the soldered components, but the **board must nevertheless be handled with care**.

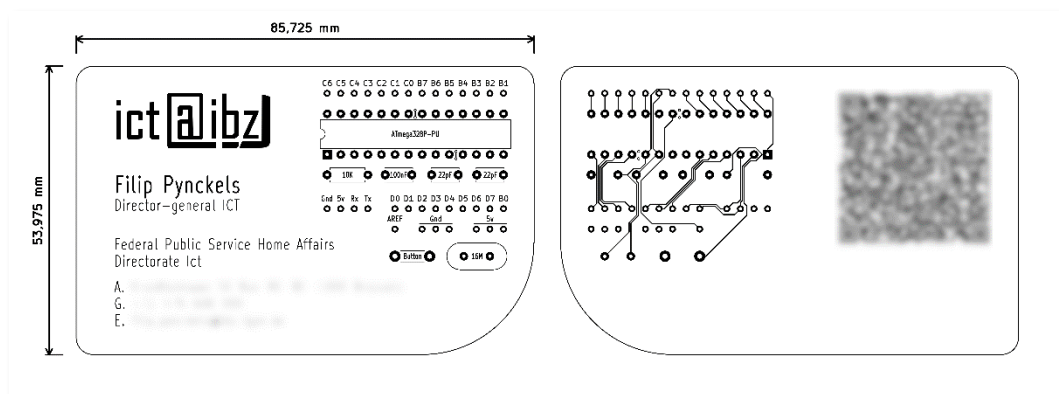


FIGURE 3 - PRINTED CIRCUIT BOARD (PCB)

The board characteristics are the following:

- 2 layers
- All traces are on the back layer
- Front layer contains Gnd filled zone
- Back layer contains Vcc filled zone
- Gnd and Vcc pins are connected to the filled zones
- 1 oz copper weight
- 0.4mm board thickness
- 0.2mm trace width
- 0.127mm trace clearance
- PCB color: blue
- Silkscreen color: white
- Surface finish: Leadfree HASL-RoHS (environment friendly, good quality)

The routing of the traces has been done manually, and care has been taken not to put traces under the crystal, nor close to the clock traces. Also, traces have been routed to ensure that the filled zones are as large as possible to provide a maximal capacitive effect.

3. Bill of materials

Table 1 shows the components that are needed to assemble a fully functional Visitekaarduino.

Processor	U1	ATmega328P-PU	Required
Capacitors	C2	47 nF	Advised
	C3	22 pF	Required with crystal
	C4	22 pF	Required with crystal
Connector	J1	1x4 male header 2.54mm	Advised
	J2	1x3 male header 2.54mm	Optional
	J3	1x3 male header 2.54mm	Optional
Crystal	Y1	16 Mhz	Optional
Resistor	R1	10 K Ω	Required
Switch	B1	1x2 pins switch 6mm	Optional

TABLE 1 - VISITEKAARDUINO BILL OF MATERIALS

Care has been taken to mark all component on the PCB with the value they must have instead of with the part number. We have done this to facilitate the assembly for non-experts. The connector pins on the other hand have been labeled so that mistakes can be avoided during use of the device.

4. Programming the ATmega328

4.1. Hardware

Figure 4 shows the [pinout of the ATmega328](#). The pins that are of importance to program the ATmega328 are the serial pins.

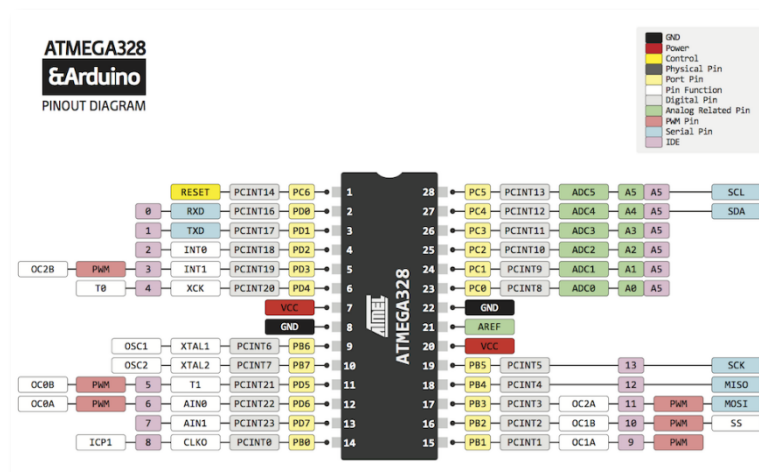


FIGURE 4 - ATMEGA328 PINOUT

For an [ATmega328 without a bootloader](#), the pins RESET/MISO/MOSI/SCK/VCC/GND form the ISP (in-system programming) interface. Figure 5 shows the ISP connector configuration.

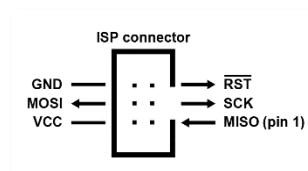


FIGURE 5 - ISP INTERFACE

For a more professional way of programming the Visitekaarduino (read: more complex, more optimization and debugging methods), one can use [Atmel Studio](#) with e.g. the [Atmel ICE programmer](#). This way of programming will make you 150€ lighter. For a more down to earth way of programming the Visitekaarduino, one can also use the [Arduino IDE programming environment with the Arduino ISP R3 card](#) (see [figure 6](#)). This setup will cost you around 12€. Unfortunately, the Arduino ISP R3 is retired. This is unfortunately, since this little piece of hardware permits you to write a bootloader to an ATmega328 chip and also permits you to load a program, written and compiled in the Arduino IDE, on the chip.

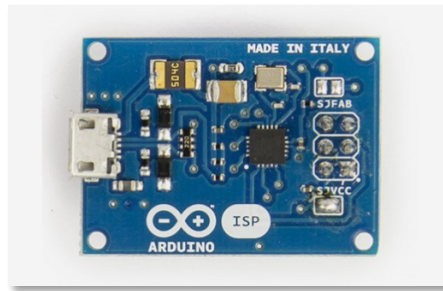


FIGURE 6 – ARDUINO ISP PROGRAMMER

One can, however, also [use another Arduino to program our Visitekaarduino](#). In that case, one can connect the programming Arduino with jumper wires to the Visitekaarduino (pins C6, B4, B3, B5, 7 and 8). Such a configuration is shown in *figure 6*. Note that the breadboard (the white board at the left) can be entirely replaced by the Visitekaarduino to program the latter, if the wires between the devices are connected as shown in *figure 6*.

For an [ATmega328 with a bootloader](#), the programming can be even simpler. In that case, just pull out the ATmega328 from an Arduino UNO and replace it with your ATmega328. Program the IC and put it in a socket on the Visitekaarduino.

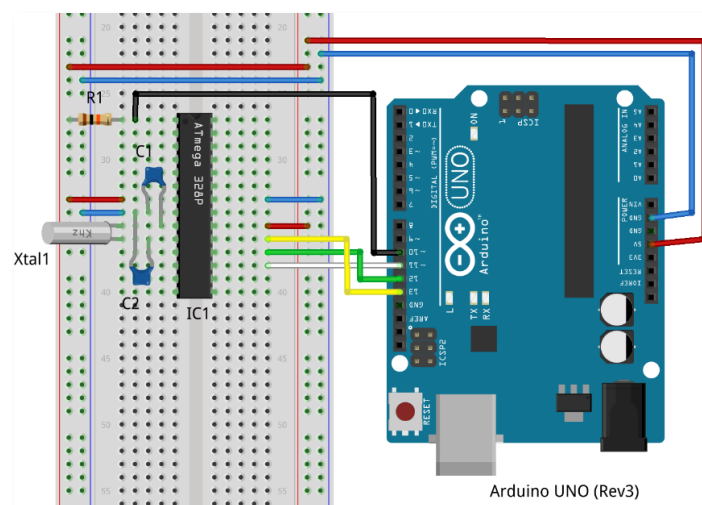


FIGURE 7 - PROGRAMMING AN ATMEGA328 WITH AN ARDUINO UNO

If you do not possess an Arduino UNO, but you have an [ATmega328 with a bootloader](#), an [Arduino USB2Serial connector](#) will do the trick. In the Arduino IDE, you chose Arduino UNO as board and. You connect the pins as follows and you're in business:

- Gnd -> Gnd
- +5V -> 5V
- Tx -> Rx
- Rx -> Tx
- Reset -> 100nF capacitor -> Reset

4.2. Software

There are two major development methods/environments to program an ATmega328 and hence the Visitekaarduino. The first development environment is the Arduino IDE, the second method is the Atmel Studio. Both have their advantages and disadvantages.

4.2.1. Arduino IDE

The [Arduino IDE](#), shown in *figure 7*, is the most user-friendly development environment. It is an open source package written in java and based on Processing which is another open source package.

Arduino IDE is easy to install, lightweight on used disk space, and comes with a plethora of pre-written code (examples, usable objects, ...).



FIGURE 8 - ARDUINO IDE - BLINK A LED PROGRAM

The programming language is very similar to a mix of C and C++. One can use both, but due to the availability of a large number of “objects” that are included by default in your program, it feels easier than C++. The used code can be found in *Appendix 1*.

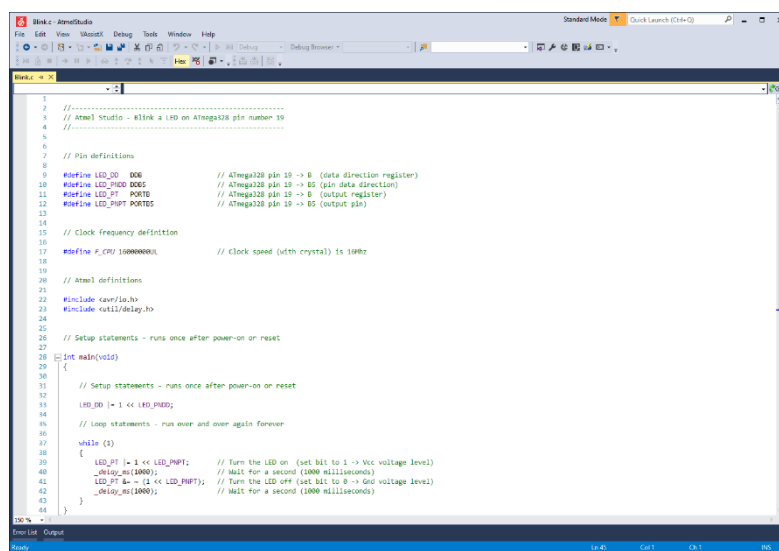
Since using the Arduino IDE and the Arduino devices are very well integrated (who would have guessed), the easiest way to program the Visitekaarduino, or any ATmega IC for that matter, is by using an Arduino device like the Arduino UNO, as shown in *figure 6*. In the “tools” menu, one can choose “Arduino as ISP” as an ISP programmer, and Bob’s your uncle.

The site where the Arduino IDE software can be found, together with a vast amount of documentation is <http://arduino.cc>.

4.2.2. Atmel Studio

The **Atmel Studio**, shown in *figure 8*, is the most complete and versatile development environment. It is a package written as an extension of Visual Studio and free to download.

Atmel Studio is easy to install. It uses a substantial amount of disk space and comes with some pre-written code (examples). If you are someone who likes to use a lot of library code to speed up your developments, then this is not the development environment you want to use. On the other hand, if you like to be able to intervene on all levels of the development, if you like to have absolute control of the generated code, if you like to use some assembly code, and especially if you like to debug your code on a controller simulation or on the real controller, then this is definitely the development environment you want to use.



```
1 //-----
2 // Atmel Studio - Blink a LED on ATmega328 pin number 10
3 //-----
4
5
6
7 // Pin definitions
8
9 #define LED_D0 DDB // ATmega328 pin 10 -> B (data direction register)
10 #define LED_P0 DDBS // ATmega328 pin 10 -> B5 (pin data direction)
11 #define LED_P1 PORTD // ATmega328 pin 10 -> B (output register)
12 #define LED_P2 PORTD // ATmega328 pin 10 -> B5 (output pin)
13
14
15 // Clock frequency definition
16
17 #define F_CPU 1000000UL // Clock speed (with crystal) is 100KHz
18
19
20 // Atmel definitions
21
22 #include <avr/io.h>
23 #include <util/delay.h>
24
25 // Setup statements - runs once after power-on or reset
26
27 int main(void)
28 {
29     // Setup statements - runs once after power-on or reset
30
31     LED_D0 |= 1 << LED_P0;
32
33     // Loop statements - run over and over again forever
34
35     while (1)
36     {
37         LED_P1 |= 1 << LED_P1; // Turn the LED on (set bit to 1 -> Vcc voltage level)
38         _delay_ms(1000); // Wait for a second (1000 milliseconds)
39         LED_P1 &= ~(1 << LED_P1); // Turn the LED off (set bit to 0 -> Gnd voltage level)
40         _delay_ms(1000); // Wait for a second (1000 milliseconds)
41     }
42 }
43
44
```

FIGURE 9 - ATMEL STUDIO - BLINK A LED PROGRAM

The programming languages you can use and intermingle are assembly, C and C++. The availability of multiple tools to debug and even profile your code makes Atmel Studio the environment for a more professional way of developing. The used code can be found in *Appendix 2*.

When you like to use Atmel Studio, a professional ISP programmer (like the Atmel ICE) are strongly advised. These kinds of programmers permit you to use the full flavor of the debugging environment in Atmel Studio, and a lot more features in general.

The site where the Atmel Studio software can be found, together with lots of documentation is <https://www.microchip.com/mplab/avr-support/atmel-studio-7>. Not that, contrary to Arduino Studio, Atmel Studio can only be used in a Windows environment, since it is built upon Visual Studio.

Conclusion

Figure 9 shows the final business card in real size (85mm × 54mm). The card thickness is 0.4mm, which makes it the size of a smartcard, but gives it the thickness of thin cardboard.



FIGURE 10 - FINISHED VISITEKAARDUINO

We refer to *figure 1* for a picture of the populated card. Populating the card will make you at most 5€ lighter.

When putting the processor in low energy mode, the controller takes something like 40μA (with clock speeds of 4Mhz). Do not forget to consider the components that are added to this card and also fed by the same power source. On average, this card without other components can run fulltime for 200 days on a 3-volt CR battery.

We have tested the card thoroughly and have found no quirks worth mentioning. We will add supplementary appendices to this text. *Appendix 3* shows how you can create a [fire, smoke and temperature warning system](#) with this card and a couple of cheap sensors. *Other appendices may follow with other projects.*

The example shows that this card is powerful enough to be at the center of a lot of [IoT \(Internet of Things\)](#) or [other projects](#), since it has the same possibilities as an Arduino UNO, but fits into smaller enclosures due to its reduced size and weight.

Appendix 1 - Arduino IDE - Source

```
//-----  
// Arduino IDE - Blink a LED on ATmega328 pin number 19  
//-----  
  
// Pin definitions  
  
#define LED 13 // Arduino UNO Pin 13 (PB5) is ATmega328 pin 19  
  
// Setup function - runs once after power-on or reset  
  
void setup() {  
    pinMode(LED, OUTPUT);  
}  
  
// Loop function - runs over and over again forever  
  
void loop() {  
    digitalWrite(LED, HIGH); // Turn the LED on (HIGH is the Vcc voltage level)  
    delay(1000); // Wait for a second (1000 milliseconds)  
    digitalWrite(LED, LOW); // Turn the LED off (LOW is the Gnd voltage level)  
    delay(1000); // Wait for a second (1000 milliseconds)  
}
```

Appendix 2 - Atmel Studio - Source

```
//-----
// Atmel Studio - Blink a LED on ATmega328 pin number 19
//-----

// Pin definitions

#define LED_DD DDB // ATmega328 pin 19 -> B (data direction register)
#define LED_PNDD DDB5 // ATmega328 pin 19 -> B5 (pin data direction)
#define LED_PT PORTB // ATmega328 pin 19 -> B (output register)
#define LED_PNPT PORTB5 // ATmega328 pin 19 -> B5 (output pin)

// Clock frequency definition

#define F_CPU 16000000UL // Clock speed (with crystal) is 16Mhz

// Atmel definitions

#include <avr/io.h>
#include <util/delay.h>

// Setup statements - runs once after power-on or reset

int main(void)
{
    // Setup statements - runs once after power-on or reset

    LED_DD |= 1 << LED_PNDD;

    // Loop statements - run over and over again forever

    while (1)
    {
        LED_PT |= 1 << LED_PNPT; // Turn the LED on (set bit to 1 -> Vcc voltage level)
        _delay_ms(1000); // Wait for a second (1000 milliseconds)
        LED_PT &= ~ (1 << LED_PNPT); // Turn the LED off (set bit to 0 -> Gnd voltage level)
        _delay_ms(1000); // Wait for a second (1000 milliseconds)
    }
}
```

Appendix 3 – A fire-smoke-temperature warning system

This appendix shows that the Visitekaarduino can be used for all kinds of projects. In this case, an [alarm system that monitors fire, smoke and temperature](#). The system sounds an alarm when parameters are out of bound.

Nota that we don't go further than a [proof of concept](#). However, as the comments on top of the source code mention, for production purposes, the used sensor breakout boards can be replaced with sensors that are attached immediately to the Visitekaarduino. The same source code can be used as for the proof of concept. *Figure 10* shows the proof of concept hardware. Below, you can find the used source code. The [Arduino IDE](#) was used as development environment.

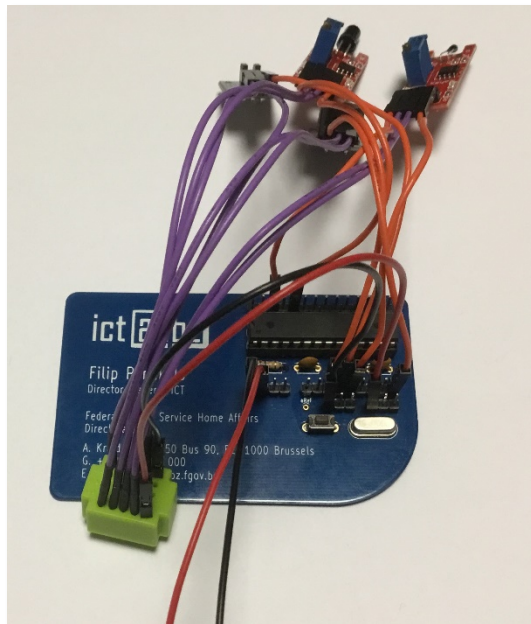


FIGURE 11 - FIRE-SMOKE-TEMPERATURE ALARM (POC)

For a [production environment](#), all the wires, the breadboard and breakout boards can be replaced by a small PCB of 15mm × 15mm or smaller that can be glued at the lower left of the Visitekaarduino. The little green breadboard in *figure 10* serves as power distribution component. The used breakout boards for the proof of concept are:

- Buzzer..... KY-006 [Passive Buzzer Module](#)
- Flame sensor..... KY-026 [Flame Sensor Module](#)
- Smoke sensor..... KY-010 [Photo interrupter module](#)
- Temperature sensor KY-028 [Digital Temperature Sensor Module](#)

On the next page, we show the source code. Note that this code can easily be extended for the use of additional sensors. The places where code must be added are mentioned in comments in the source code.

```

//=====
//
// Project: Visitekaarduino_Danger_Alert
// File   : Visitekaarduino_Danger_Alert.ino
// Version: 1.0
//
// Author : Filip Pynckels
//
// Created: 2019 02 10
//
// Build tools:
//   Arduino IDE
//   Buzzer      - KY-006 Passive Buzzer Module
//   Flame sensor - KY-026 Flame Sensor Module
//   Smoke sensor - KY-010 Photo interrupter module
//   Temperature sensor - KY-028 Digital Temperature Sensor Module
//   Visitekaarduino
//
// Production tools:
//   Battery      - CR2032
//   Buzzer      - Passive piezoelectric buzzer (1.5kHz ~ 2.5kHz)
//   Flame sensor - 5mm infra-red receiver LED (760 nm to 1100 nm)
//   Gas & Smoke sensor - MQ2, MQ5, MQ9
//   Temperature sensor - NTC thermistor (-55°C to 125°C)
//   Visitekaarduino
//
// Serial port setting:
//   Ascii,9600,8,N,1
//
// Remarks:
//   For production purposes, chose a buzzer that generates the necessary
//   decibels. Drive this
//=====

//
//           Arduino UNO pin      Visitekaarduino pin      Atmega328P pin
//           -----
int BUZZER      = 8;              //      B0              14      // KY-006 buzzer pin
int FLAME_DIGITAL_PIN = 2;        //      D2              4      // KY-026 flame sensor pins
int FLAME_ANALOG_PIN  = A2;       //      C2             25
int SMOKE_DIGITAL_PIN = 3;        //      D3              5      // KY-026 smoke sensor pin
int TEMP_DIGITAL_PIN  = 4;        //      D4              6      // KY-026 smoke sensor pins
int TEMP_ANALOG_PIN   = A4;       //      C4             27

//
// --- More sensor pins can be added here ---
//

int FLAME_SOUND      = 1;              // Set buzzer wavelength (ms) indicators
int SMOKE_SOUND      = 2;
int TEMP_SOUND       = 3;

//-----

void logAlert(String message)
{
    Serial.println("ALERT ----- " + message);
}

void logCall(String message)
{
    Serial.println("Call - " + message);
}

void logErr(String message)
{
    Serial.println("Error - " + message);
}

void logMsg(String message)
{
    Serial.println("Mesg - " + message);
}

//-----

void buzzerSetup()
{
    logCall("buzzerSetup()");

    pinMode(BUZZER, OUTPUT);          // Set buzzer pin as output
}

void buzzerSound(int wavelength)
{
    logCall("buzzerSound(" + String(wavelength) + ")");

    for (int i = 0; i < 80; i++)
    {
        digitalWrite(BUZZER, HIGH);    // Send high signal to buzzer
        delay(wavelength);             // Delay in ms
        digitalWrite(BUZZER, LOW);     // Send low signal to buzzer
        delay(wavelength);             // Delay in ms
    }
}

//-----

```

```

void flameSetup()
{
    logCall("flameSetup()");
    pinMode(FLAME_DIGITAL_PIN, INPUT);
    pinMode(FLAME_ANALOG_PIN, INPUT);
}

bool flameSensed()
{
    logCall("flameSensed()");

    int digitalFlame = digitalRead(FLAME_DIGITAL_PIN);
    logMsg("    digitalFlame = " + String(digitalFlame));

    int analogFlame = analogRead(FLAME_ANALOG_PIN);
    logMsg("    analogFlame = " + String(analogFlame));

    return ((digitalFlame == HIGH) || (analogFlame < 500));
}

//-----

void smokeSetup()
{
    logCall("smokeSetup()");
    pinMode(SMOKE_DIGITAL_PIN, INPUT);
}

bool smokeSensed()
{
    logCall("smokeSensed()");

    int smoke = digitalRead(SMOKE_DIGITAL_PIN);
    logMsg("    smoke = " + String(smoke));

    return (smoke == HIGH);
}

//-----

void tempSetup()
{
    logCall("tempSetup()");
    pinMode(TEMP_DIGITAL_PIN, INPUT);
    pinMode(TEMP_ANALOG_PIN, INPUT);
}

bool tempSensed()
{
    logCall("tempSensed()");

    int digitalTemp = digitalRead(TEMP_DIGITAL_PIN);
    logMsg("    digitalTemp = " + String(digitalTemp));

    int analogTemp = analogRead(TEMP_ANALOG_PIN);
    logMsg("    analogTemp = " + String(analogTemp));

    return ((digitalTemp == HIGH) || (analogTemp < 500));
}

//-----

void setup()
{
    Serial.begin(9600);
    while (!Serial) { }
    logCall("setup()");

    buzzerSetup();

    flameSetup();
    smokeSetup();
    tempSetup();

    //
    // --- More sensor setup calls can be added here ---
    //
}

//-----

bool firstLoop = true;

void loop()
{
    if (firstLoop)
    {
        firstLoop = false;
        logCall("loop()");
    }

    Serial.println("\n");

    if (flameSensed())
    {
        logAlert("FLAMES");
        buzzerSound(FLAME_SOUND);
    }

    Serial.println("\n");

    if (smokeSensed())

```



```

    {
        logAlert("SMOKE");
        buzzerSound(SMOKE_SOUND);
    }

    Serial.println("\n");

    if (tempSensed())
    {
        logAlert("TEMPERATURE");
        buzzerSound(TEMP_SOUND);
    }

    //
    // --- More sensor routine calls can be added here ---
    //

    delay(1000);
}
// Second priority sound
// Third priority sound
// Wait 1s before repeating loop()

```