



Arduino based Data Acquisition System

Final Project

15/04/2016

Project Supervisor: Assoc. Prof. Dr. G. Nikolov

Student: Megara Veramendi Fernández

Bachelor degree: Technical Engineering in Industrial Technologies

Index

1. Precedents	3
1.1. Presentation.....	3
1.2. Aim of the project.....	4
1.3. Reach of the project	5
2. Software and Hardware involved	6
2.1. Arduino Uno	6
2.1.1 Specifications for Arduino UNO.....	7
2.1.2 Power supply, inputs and outputs.....	8
2.1.3 Digital inputs and outputs.....	8
2.1.4 Analog inputs.....	9
2.1.5 Basic functions.....	10
2.1.6 Program structure.....	11
2.1.7 Functions.....	12
2.1.8 Syntax.....	12
2.1.9 Variables.....	13
2.1.10 Arithmetic.....	13
2.1.11 Boolean constants. Conditional systems.....	14
2.1.12 Iterative systems.....	14
2.2. LabVIEW.....	16
2.2.1 Virtual instrument VI.....	16
2.2.2 Software structure.....	17
2.2.3 Work environment.....	17
2.2.4 Program creation.....	26
2.3. LINX	27
2.4. Using ARDUINO through LabVIEW.....	27
3. Sensor programming.....	31
3.1 Temperature sensor KY-013.....	32
3.2 Light Sensor. Phototransistor TEMT6000.....	39
3.3 Laser Diode Module KY-008.....	44
3.4 PIR sensor HC-SR501.....	48
3.5 Magnetic Hall sensor KY-024.....	53
3.6 Wireless communication. XBee modules.....	56
4. Sensor functioning	61



5. Applications or Uses.....	63
6. Budget.....	66
7. Conclusions.....	67
8. Bibliography.....	68
Appendix I:	70
Appendix II:	72
Appendix III:	79

1. Precedents

Below, an introduction to the project that is going to be explained in these documents is going to be presented; the topic is **Arduino based data acquisition system**.

1.1 PRESENTATION

The project which is presented in these documents about the design of an Arduino based data acquisition system is carried out under the supervision and guidance of **Prof. Dr. G. Nikolov**, from the Faculty of EE and IT of the Technical University of Sofia.

The documents here presented are the details of the design and development of a Measurement, Monitoring and Control System using the software from **Arduino**. This system is developed by using the software provided by **LabVIEW**. LabVIEW will help to have a system with an easier programming thanks to the interface it offers.



In order to try this system, different sensors will be tested and used so that different parameters can be measured and processed.

During this project, the most important points regarding the system are gathered, which will allow the reader to have vision on how it works. These points are mainly the description on how Arduino and LabVIEW software work, how they can interact with each other, and how to use them to program a measurement, monitoring and control system.

So that the goal of the project can be accomplished, different books and Internet information have been used as bibliography, along with the advices and help of Prof. Dr. G. Nikolov.

1.2 AIM OF THE PROJECT

As it has been introduced before, the **main objective** of this project is to design and implement a system which can measure and process different parameters such as temperature or light by using the Arduino platform and LabVIEW software, with the corresponding tests.

It is expected that the system allows the user to control in a very easy way the microcontroller by using a computer, with no need of having a deep knowledge about programming in Arduino. The user will be able to read the information and control the system by using LabVIEW's interface, which is intuitive and comfortable to use.

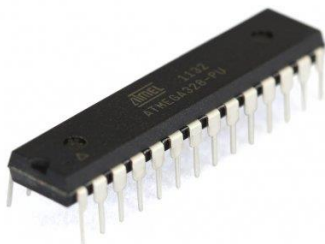
The way on how the system measures and controls the parameters is one of the main goals. The user of the system cannot receive any wrong data, or data with poor precision, because that would mean a failure on its application. Therefore, no mistakes are allowed as it can affect the process in which this system may be used.

Moreover, one more objective is that once the sensors and microcontrollers are installed, it is easy to access and get the information the sensors provide for the user. This means that you will not need to be an expert on programming if you want to use it. LabVIEW interface will help with this objective. In one word, **intuitive**.

Versatility in this project will also be one of the characteristic of this Arduino based data acquisition system. It will be possible to measure and process several parameters, not only a specific one and above everything, be useful for general purposes.

Basically, the components that form part of the system are:

- Arduino Uno microcontroller
- Different sensors
- Arduino software
- LabVIEW software
- Computer



1.3 REACH OF THE PROJECT

The reach of this project is going to be the design and development of a prototype of an Arduino based data acquisition system by using LabVIEW in order to program and control the system.

The sensors that are going to be used in the system are basic elements suitable for learning and try them on the prototype. The aim of the project is not to use this system in any specific industrial process, but to learn and show how to design it, and be able to measure different parameters. The interest is not in designing difficult circuits, but in discover how useful is to use a microcontroller with a visual interface in the computer.

However, this project could perfectly be the base for a more complex design that could be used in any industrial process, such as the temperature control of an oven.

In conclusion, the purpose of this project is more educational, although it is the base for industrial applications.

2. Software and hardware involved

Throughout this chapter, the different software used in the project are going to be presented, and it is going to be explained how they work and explore their possibilities and applications. The software described in this chapter is:

- Arduino
- LabVIEW
- LINX

2.1 ARDUINO UNO

Arduino is a physical computing platform with open code, based on a board with a simple microcontroller and a development environment to create software (programs) for the board.

You can use Arduino to create interactive objects, reading data from a great variety of switches and sensors, and control different kind of lights, motors and other types of physical actuators. Arduino projects can be autonomous or they can communicate with other software running on a computer (such as LabVIEW). You can design your own board, or even buy it already prepared and ready to use. Software is completely free and can be downloaded from the Internet.

Arduino can be powered via USB connection or with an external power supply. The power supply is being selected automatically.

There are different versions of this microcontroller: Arduino UNO, Arduino Leonardo, Arduino Pro, etc. In the project, **Arduino UNO will be used.**



2.1.1 Specifications for Arduino UNO

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current for I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328)
SRAM	2 KB (ATmega328)
EPROM	1 KB (ATmega328)
Clock Speed	16 MHz

2.1.2 Power supply, inputs and outputs

Either Arduino is supplied with USB connection or with an external power supply (recommended with 7-12V), outputs are going to have a continuous voltage due to voltage regulators and stabilization capacitors present on the board. These power supply pins are:

- **VIN:** it is the input power supply that will have the same voltage that we are supplying the Arduino with the external power supply
- **5V:** power supply of 5V, this voltage may come from VIN pin and a voltage regulator or from the USB connection.
- **3.3V:** power supply that will provide 3.3V generated by an internal regulator, with a maximum current of 50 mA.
- **GND:** grounding pins

2.1.3 Digital inputs and outputs

Each of the 14 digital pins can be used as an input or output. Besides, each pin can supply or receive a maximum of 40 mA and has a pull-p resistance from 20 to 50 kOhm. In addition, some pins have specialized functions such as:

- Pin 0 (RX) and 1 (TX). They are used to receive (RX) and transmit (TX) in TTL serial communication.
- Pin 2 and 3. External interruptions. Pins in charge of interrupting the sequential program established by the user.
- Pin 3, 5, 6, 9, 10 and 11. PWM (pulse width modulation). They form 8 output bits with PWM with the function `analogWrite ()`.
- Pin 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI (Serial peripheral interface) communication.

- Pin 13. LED. There is a LED connected to the digital pin 13. When the pin value is HIGH, the LED is on, and when the value is LOW, the LED is off.



Figure1. Digital inputs and outputs

2.1.4 Analog inputs

Arduino UNO has 6 analogical inputs, from A0 to A5, and each one offers a resolution of 10 bits (1024 states). By default, there is a voltage of 5V, but that range can be modified using the pin AREF and using the function `analogReference()`, where the user introduces the DC external signal he wants to use as reference.

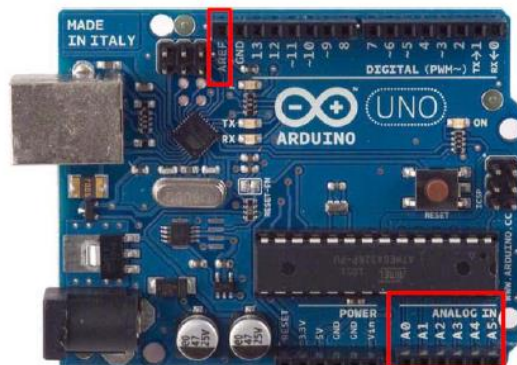


Figure 2. Analog inputs

2.1.5 Basic functions

It is important to know the basic functions of Arduino software although in this project, programming directly with Arduino is not one of the objectives.

- **Digital I/O:**

pinMode(pin, mode)

It configures the specific pin the user wants as an input or output.

digitalWrite(pin, value)

It assigns the HIGH value (5V) or the LOW value (0V) to a digital pin.

digitalRead(pin)

It reads the value of a specific pin, HIGH or LOW.

- **Analog I/O**

analogRead(pin)

It reads the voltage value in the analog pin. Arduino board has 6 channels connected to an analog to digital converter of 10 bits. This means that voltages between 0 and 5V will be transformed into a number from 0 to 1023 units, or what is the same, 4.9 mV per unit. The input range can be modified using the function *analogReference()*.

analogWrite(pin, value)

It gives an analog (PWM) value on a pin. This may be used to control for example the light intensity from a LED or the speed of a motor. The signal given by the pin is a square signal with a specific duty cycle chosen by the user. The frequency of the PWM signal is around 490 Hz. The possible values to configure the duty cycle go from 0 to 255.

- **Serial communication**

It is used for the communication between Arduino and a computer or other devices. Every Arduino board has at least one serial port. This ports communicates thanks to the digital pins 0 (RX) and 1 (TX), and with the computer thanks to the USB connection.

Serial.begin(speed)

It establishes the speed of data in bits per second (bauds) for the transmission of data in serial communication.

Serial.read()

It reads the data from the serial port.

Serial.print(val,[format])

It prints the data to the serial port as ASCII text

Serial.println(val,[format])

It prints the data to the serial port as ASCII text but it jumps to a new line.

Serial.available()

It gives back the number of available bytes to be read by the serial port. It refers to data that has already been received and is available in the buffer of the port.

2.1.6 Program structure

The basic structure of programming language in Arduino is quite simple, and is made up at least with two differenced parts. These two necessary parts, or functions, enclose blocks that have different statements or instructions.

```
void setup()
{
  instructions;
}
void loop()
{
  instructions;
}
```

It is in **setup()** where you find the part in charge of gathering the configuration, and in **loop()** you find the instructions of the program that will be executed over and over again in a loop. Both of these parts are indispensable so that the program works fine.

The function of the configuration `setup()` must have the definition of the variables. It is the first part that is going to be executed, only one time, and is used to configure or initialize `pinMode`, configuration of serial communication and others.

```
void setup()
{
  pinMode(pin, OUTPUT);
  // configures the 'pin' as output
}
```

The loop function has the code that will repeatedly be executed (reading of inputs, activation of outputs...). This is the nucleus of all programs in Arduino, and is the one that realizes most part of the work.

```
void loop()
{
digitalWrite(pin, HIGH);
delay(1000); // waits one second (1000 ms)
digitalWrite(pin, LOW); //
delay(1000);
}
```

2.1.7 Functions

A function is a piece of code that has a name and a group of instructions that are executed when the function is called. User functions can be written to do repetitive tasks and to reduce the size of the program.

These functions are associated to a type of value “type”. This value will be the one given back by the function. If the function does not return any type of data, a *void* function will be used. After declaring the type of data the function returns, its name must be written with parameters between brackets.

```
type functionname(parameters)
{
instructions;
}
```

2.1.8 Syntax

- *Keys { }*: they are used to define the start and the end of a block of instructions.

```
type funcion()
{
instructions;
}
```

- *Semicolon ;*: used to separate instructions from each other. It is also used to separate elements in instructions such as *for loops*. It is important to make sure that there is a “;” at the end of every line, or else there will be a compilation error.
- *Comments*: comments are text that are ignored by the program and are used for descriptions of the code to help in the understanding of the program. They start with a */** and end with **/*. For a single line of comments, *//* can be used at its start. They do not use space of the Arduino memory.

2.1.9 Variables

A variable is a way of naming and storing a numeric value for a following use. As its own name says, variables are numbers which can constantly change, just the opposite of constants. Variables must be declared and even assign an initial value.

It is helpful that descriptive names are given to variables, so that they are used in a very easy way, and to help other people read the code.

Every single variable must be defined at the start of the code with the type data they are going to store such as *int*, *long* or *float*. It is also necessary to write a name, and an initial value if needed.

Variables may be declared at the beginning of the program, inside functions and even inside loops. If you declare a variable at the setup () part, they're called global variables and can be used along the whole program. On the contrary, local variables can only be used in the specific functions or loops.

2.1.10 Arithmetic

Arithmetic operators that are included at Arduino programming are sum, subtract, multiplying and division.

```
y = y + 3;  
x = x - 7;  
i = j * 6;  
r = r / 5;
```

The arithmetic operations are executed depending on the type of data of the operands. This means there may be overflows if the result of the operation is bigger than the type of data can manage.

You can also find logic operators, AND (&&), OR (||) and NOT (!).

2.1.11 Boolean constants. Conditional systems

- Boolean variables: Boolean constants define the levels **HIGH** or **LOW** when they are referring to the state of digital outputs. **FALSE** is associated with 0 while **TRUE** is associated with 1 or anything different from 0.
- **If**: it is an instruction used to verify if a condition is having place or not. If the condition is TRUE, specific instructions will be executed. In case it is false, the program will jump this part of code.

```
if (variable ?? value)
{
Execute instructions;
}
```

The user can use *if... else* structure so that in case the condition is false, other instructions are executed. You can connect several *else* in a row.

2.1.12 Iterative systems

- **For**: the *for* loop is used to repeat a set of instructions a determined number of times set by the user. Every single time the instructions end, the program starts executing the same instruction inside the loop until it reaches the number of times said by the user.

```
for (initialization; condition; expression)
{
executeinstructions;
}
```

- **While**: it is a loop whose execution is active while a condition set by the user is TRUE

```
While (variable ?? value)
{
instructions;
```

- **Do... while:** this loop works the same way as the *while* loop, with the exception that the condition is being tested at the end of the loop and not at the start, so that at least this loop is executed once.

```
do  
{  
  intructions;  
} while (variable ?? value);
```

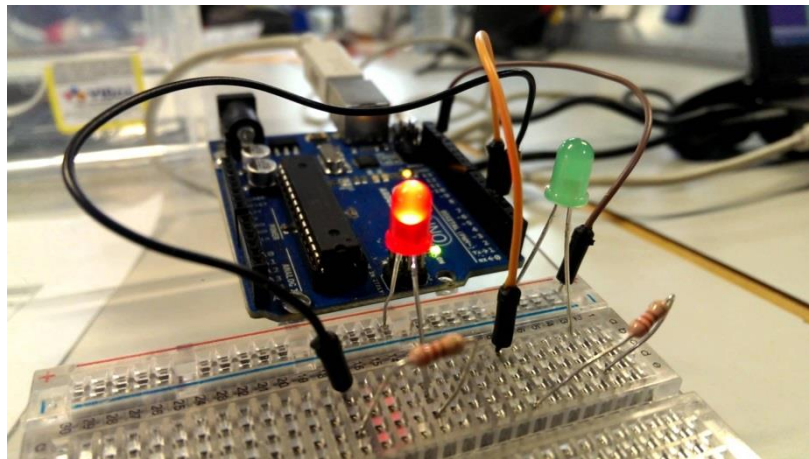


Figure 3. Example of Arduino structure with LED and resistances.

2.2 LABVIEW

LabVIEW is a high level programming language, which inherits the internal structures of C language, but offers more simplicity and functionality que it is about making signal processing calculations and control techniques of instruments are devices in general. This software is supported by National Instruments.

The main advantage that LabVIEW offers is that it is a **graphic language**. The programs in this language have, from the user side, the same appearance than the front panel of an electronic device, with buttons, graphic screens, numerical indicators, etc. LabVIEW offers the possibility of creating the same control structures with a kind of “module” look, interconnected among them with cables that carry information from ones to another. In this way, those modules or sections- called in LabVIEW virtual instruments (VI)- are the equivalent to “functions” in a classic programming language and the “cables” that interconnect would act as input or output variables of these functions.



2.2.1 Virtual Instrument VI

A traditional instrument provides as software as measurement circuits embedded in a product with a fixed functionality using the front panel instrument. On the other hand, the concept of virtual instrumentation is born from the use of the personal computer as an “instrument” used for measuring signals, acting itself as the control center for multiple measuring devices (oscilloscopes, generators, multimeters, etc.)

LabVIEW offers the possibility of emulating and control through software real electronic equipment as the one you can find in a laboratory. This software will be running in a computer that will have access to different hardware, data acquisition cards (analog and digital), interface cards with instrumentation buses and analog and digital control channels.

The virtual instruments VI will allow to handle hardware by a graphic interface that will be similar to a control panel of usual devices (oscilloscope, multimeter, etc.). With the representation in the screen of the graphic visualization elements and control

that will be used as interface with the user, this will observe the state of the selected inputs on the screen. The user will be able to interact with the outputs directly or with the execution of different routines that have been programmed.

2.2.2 Software structure

Basically, the software will be in charge of communicating the user interface of the computer with the data acquisition hardware, giving the application with the required functionality.

Several layers on this software can be distinguished: application program, drivers and application libraries (API's) as can be seen in the image.

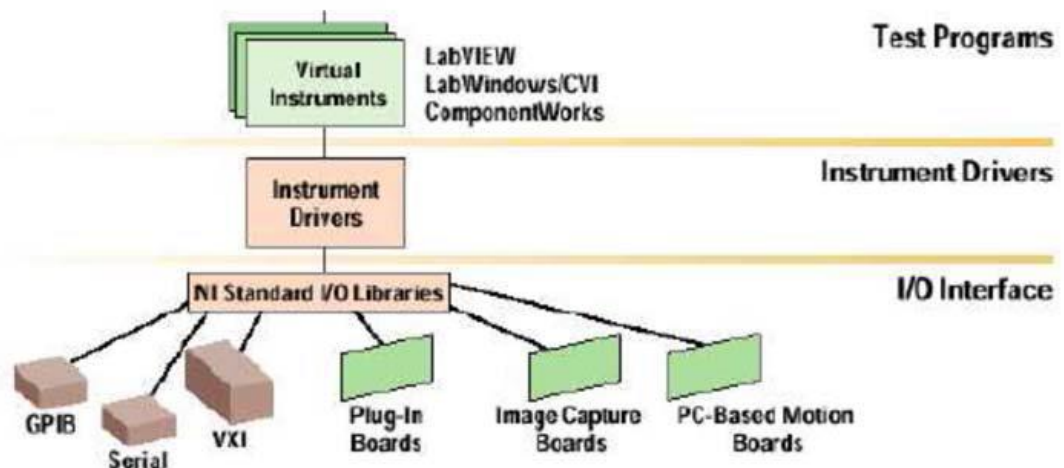


Figure 4. Software structure of LabVIEW

2.2.3 Work environment

LabVIEW is a graphic programming tool. Originally it was orientated for applications for controlling electronic instruments used in the development of measurement systems, what is known as virtual instrumentation. For this reason, programs created in LabVIEW will be saved in files called VI and with the same extension ".vi", which means Virtual Instruments.

Also related with this concept, its two main advantages are given a name: a real instrument will have a Front Panel where you can find the buttons, displays, etc. and an internal circuitry. In LabVIEW, these two parts are given the names of Front Panel and Block Diagram, respectively.

- **Front Panel:** it is the part that the user will see, it usually has a grey background.
- **Block Diagram:** It is where the programming will be made and it usually has a white background.

In order to change from the Front Panel to the Block Diagram, the user can use the command **CTRL+E**

The Front Panel and the Block Diagram are connected through terminals (elements that are used as data inputs or outputs). On the same way that a luminous indicator of an instrument is represented by a LED in the circuitry, in a LabVIEW program that same indicator will be represented in the Block Diagram as a Boolean output on which to write a value.

In the next figure, both of these parts can be seen. The panel on the left is the Front Panel, with grey background, a graphic and a terminal: a graphic that represents the sinusoidal in the time, and a stop button. In the Block Diagram, we can see these two elements too under the same names and represented with two icons. Next to this, it appears a function that simulates a sinusoidal signal.

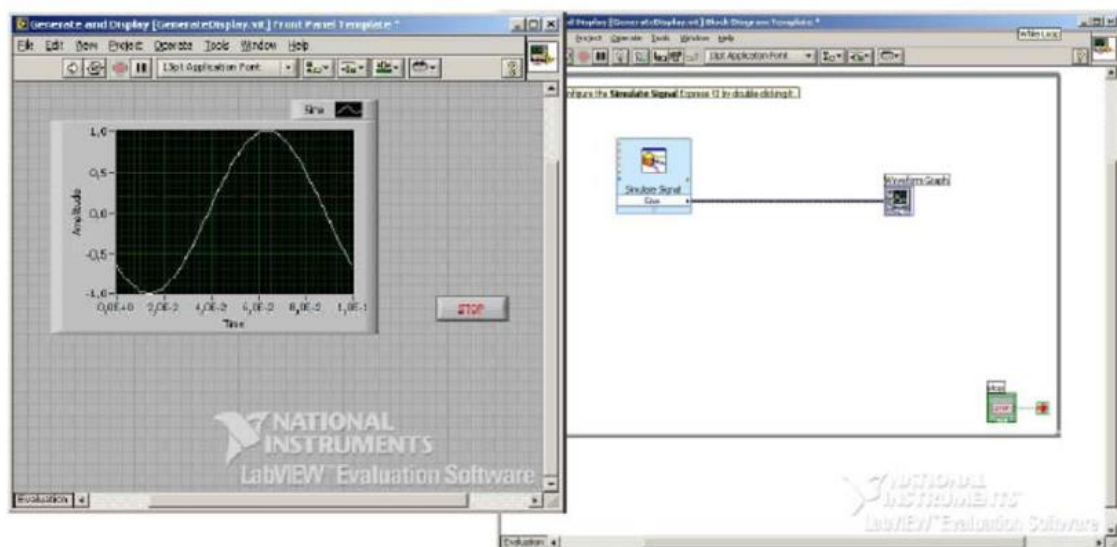


Figure 5. Front Panel and Block Diagram

In the upper part of these windows, there is the toolbar with several tools. In the Block Diagram, this bar has several more options.

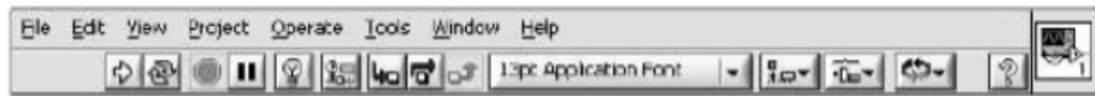


Figure 6. Toolbar.



The first group of tools is useful to control the execution of a program in LabVIEW. The first button indicates if there are errors in the program (broken arrow) and, when everything is right (full arrow like in the picture), executes once the program. The second one, executes the program in a loop, but as a general rule, this must not be used. Instead, a loop in the program has to be included. The third button aborts the execution and the fourth allows a pause.



The second group of buttons is useful to help in deputation. The first button is Highlight Execution, one of the most useful tools to deparate, slows down the execution allowing seeing the path of data. Next one, Retain Wire Values, allows obtaining the previous value when placing a probe. Three next buttons are used to execute the program step by step.



The pull-down menu allows to format texts, and is recommendable to use the predefined formats such as Application Font or System Font.



Next group to align, distribute, control the size, and group and order objects.



Last of all, in the window of the Block Diagram, exists a button that allows the automatic routing of cables. It is very recommendable to use it, because when hitting the button, the program automatically orders the scheme to make it as clear as possible.

On the right side of the Front Panel and the Diagram Block, there is an icon that represents the VI. It has a contextual menu that allows to access to different characteristics. To place functions in the Block Diagram and terminals in the Front

Panel, there are palettes, or floating menus, called functions palettes or controls palettes.

Moreover, we have the tools palette that is made up of different parts. Another very useful trick is to use CTRL+H in order to open the help window.

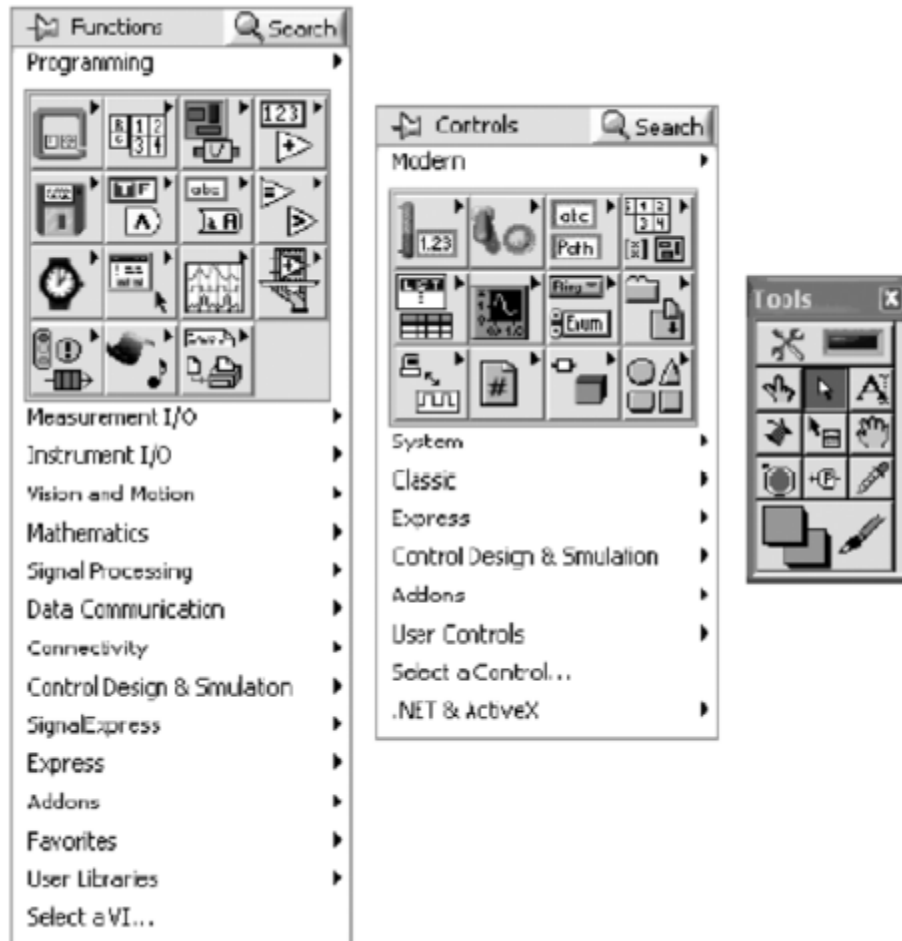


Figure 7. Different working palettes, functions, controls and tools.

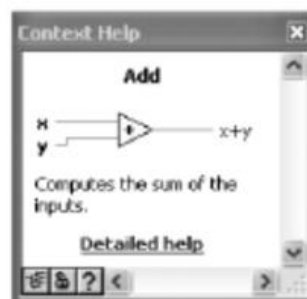


Figure 8. Contextual Help Window

- **Tools Menus:** With the tools palette you can select the instrument to be used, and this makes the mouse cursor to change.



Figure 9. Tools palette

The function of each of the tools is the next:

- Automatic Tool Selection: if activated, it automatically chooses the tool depending on the element where the cursor is.
- Operate value: the cursor available when the program is working, it is useful to change values in the controls at the Front Panel.
- Positon/Size/Select: useful to place objects and also to change sizes and select elements
- Edit Text: changes the cursor so that text can be written. It may be use to write comments in the Block Diagram.
- Connect Wire: the tool used for wiring up. By uniting element between them, allows data to flow through them. If the cables are grey and discontinuous means that the cable is broken (there is an error): type of data is incompatible or the terminals are not the right one.
- Object shortcut Menu: pulls down the contextual menu associated to an element. It is equivalent to clicking with the secondary button of the mouse.
- Scroll Window: used to move the content of windows the same way as if the side scroll bars were used.
- Set/Clear Breakpoint: Creates or deletes a breakpoint on an element. When the execution reaches this point, it stops.

- Probe Data: creates a Probe in a cable. A probe is a floating window that displays the value circulating through the cable.
 - Get Color: Obtains the color value of an element.
 - Set color: colors an element. IT has two possible colors, the main one and the background one. Both of them can be assigned independently.
- **Controls**: terminals are divided in controls and indicators, although sometimes they all are called controls. The real controls are data inputs and outputs indicators. They are classified according to their style in different submenus: *Modern*, *System* and *Classic*. Inside each submenu there are other menus that classify the controls according to the type of data. The most used are the ones to make graphics and those from the first line of submenu *Modern*.

To use the controls, the desired terminal must be selected and taken to its place on the Front Panel.



Figure 10. Control palette



Figure 11. Graphic palette

The indicators for graphics are really important in this software. *Waveform Chart* and *Waveform Graph* are the most important ones.

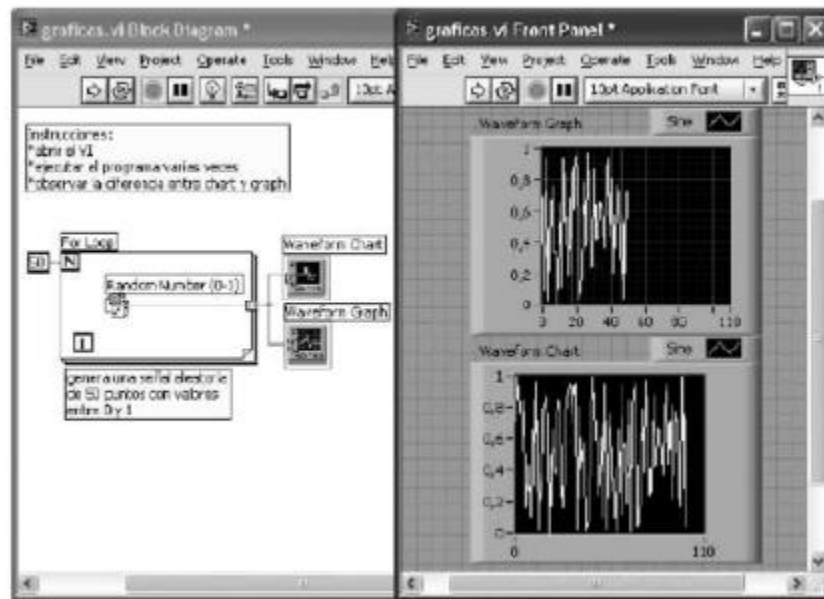


Figure 12. Comparison between Waveform Graph and Waveform Chart

The *Waveform Chart* indicators are different from *Waveform Graph* ones, because *Waveform Graph* indicators fully draw the graphic when new data arrive, however, *Waveform Chart* annex the new data to the previous ones. The number of data that are shown can be adjusted or user can use *Autoscale*. There are other features that can be adjusted on the contextual menu of each indicator.

- **Functions:** this is the palette shown when working on the Block Diagram. You can access to different functions, subVIs and other structures. As with the controls, there are also different submenus depending on the application. The most used functions are the ones from *Programming*.

In *Programming* you can find *Structures*. It has the elements that are equivalent to the control instructions in conventional languages. For example you can find loops such as WHILE, FOR, and the conditional structure CASE.

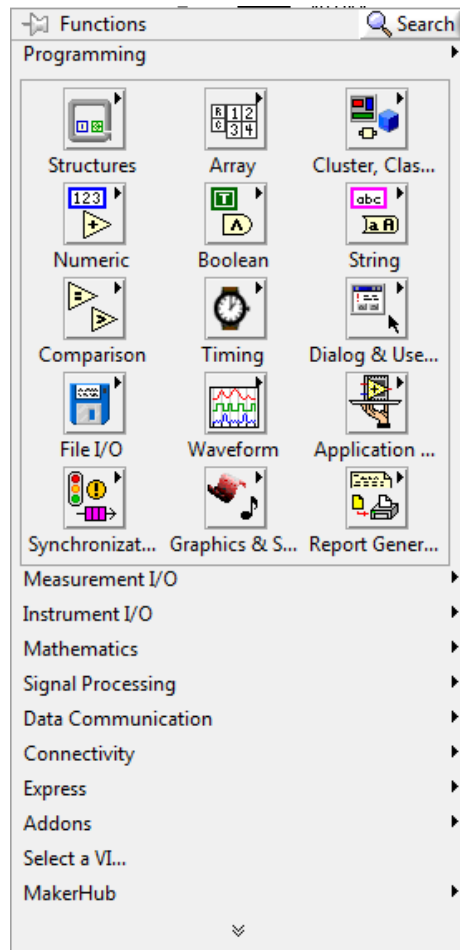


Figure 13. Functions palette

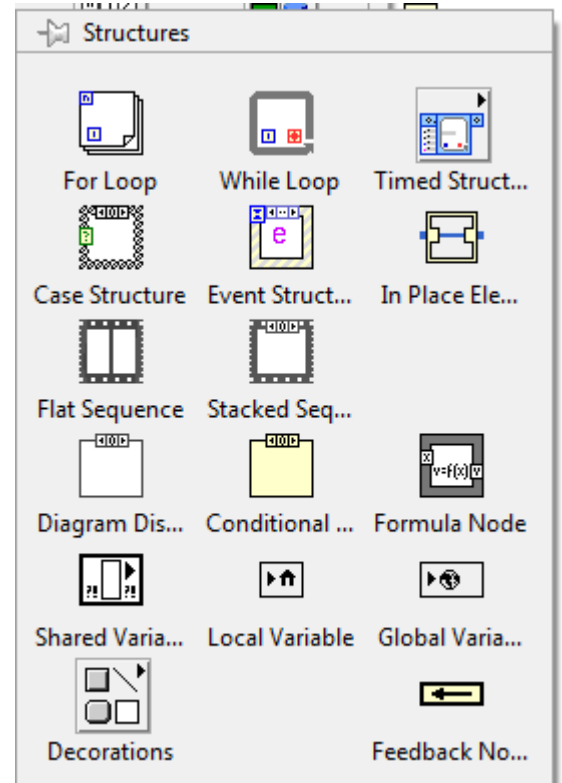
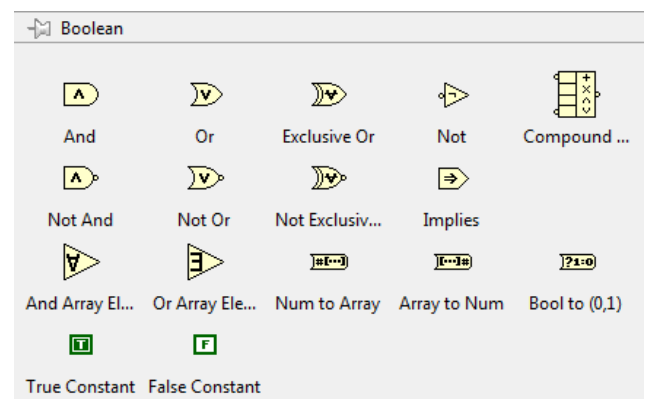
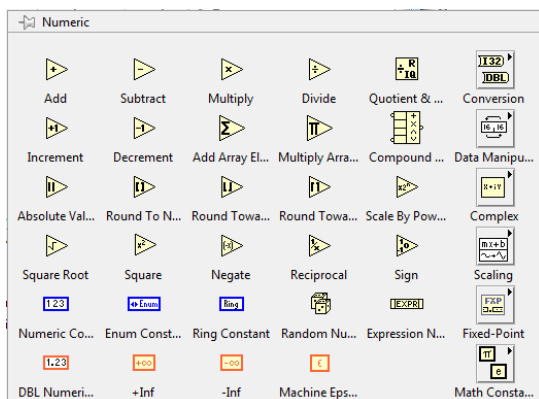
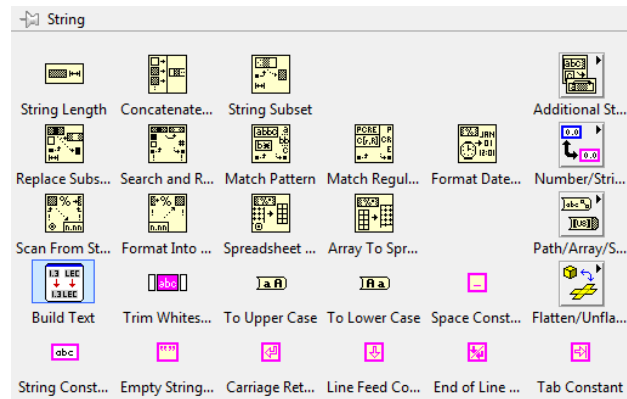


Figure 14 Structures palette

The next menus in *Programming* are divided according to the type of data. There simple data such as **numeric, Booleans and strings**, and compound ones such as clusters or arrays. Each of these menus has functions to work with data.





Example of numeric, Boolean and String palettes.

Numeric data are divided in entire or floating point, and with different sizes. By using the contextual menu > *Representation* you can change the type of number so that you operate with the same type of data. **Boolean** data can only have two values, TRUE or FALSE, so this type of data is suitable to create buttons.

Compound data are formed by a large number of individual data, but they are all the same type. Arrays are organized data while *clusters* are not organized.

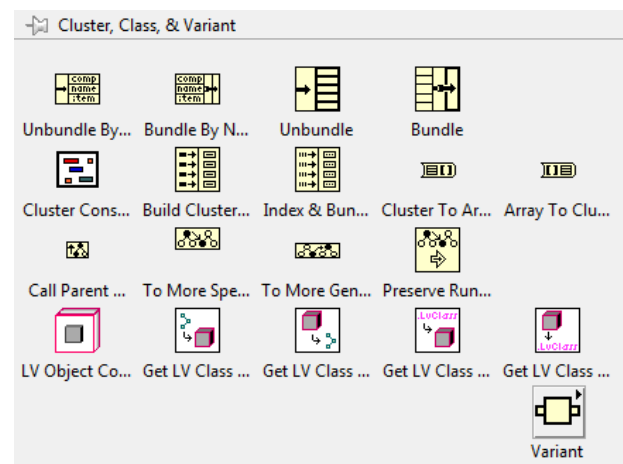
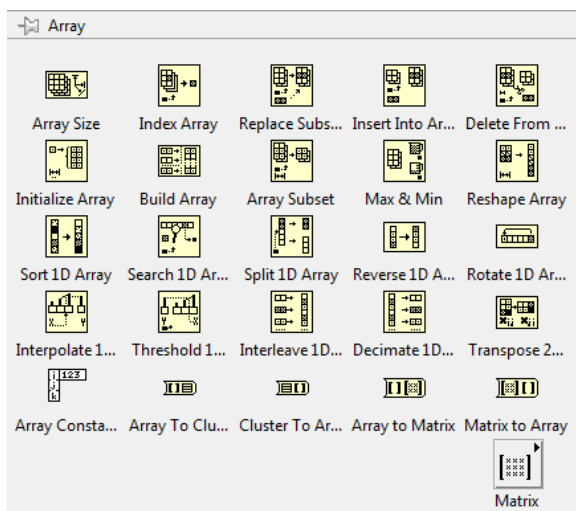


Figure 15a. Arrays and cluster palettes

The functions and Vis generally operate over one or several inputs and will return one or more outputs. The correct type of data will be connected to the input. This may come from a control, a constant or an output from another VI or function and its output can be connected to an indicator or may be the input of a different VI or function.

2.2.4 Program Creation

In LabVIEW the programming is made in the Block Diagram. A program is usually made up by:

1. Controls: they are used as inputs for data.
2. Functions, Vis and structures: they realize one or several operations with this data.
3. Indicators: they are used as output for data

Data flow around the program by cables, which are used to connect the elements between them. To make the connection, the *Connect Wire* tool may be used. A cable has a unique source (control, constant or output from other element), and one or several destinations (indicator or input of other elements), all of them compatible with the same type of data. Wiring in general should be as short as possible so that the program is clear. A very useful tool for this is *Clean Up Wire*, which realizes an automatic sketch. A good wiring sketch is not only more elegant, but also it can make the program to have higher performance regarding the used memory.

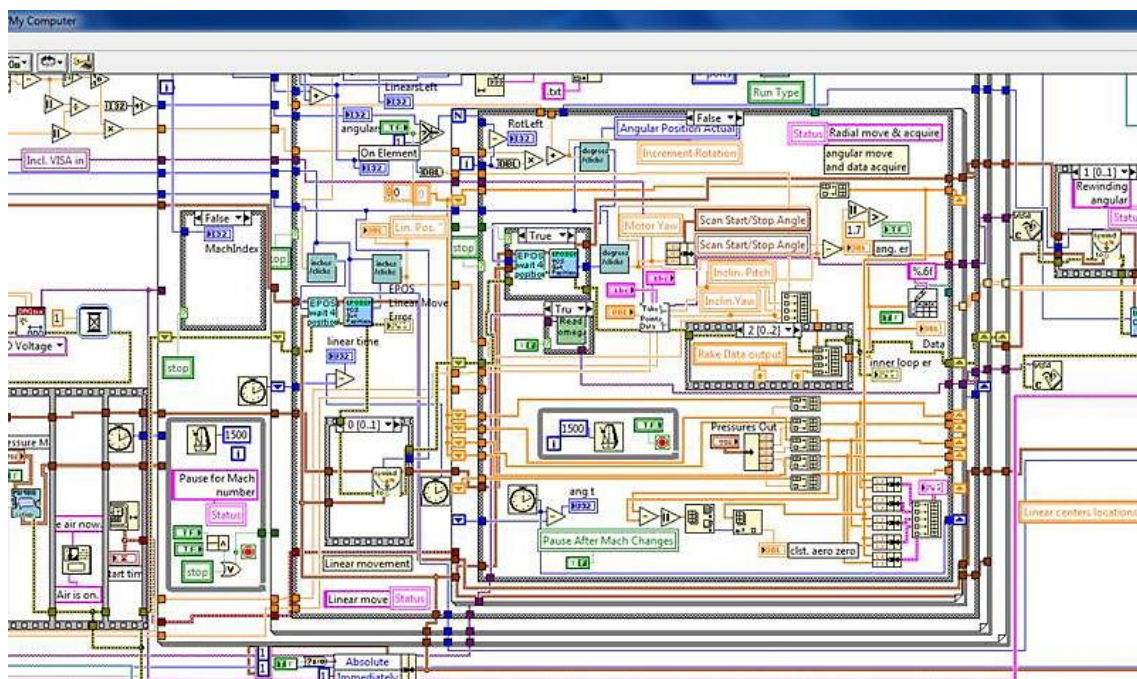


Figure 15b. Example of a complex program using LabVIEW

2.3 LINX

Linx is a package installed in LabVIEW with the help of *VI Package Manager*. It is free software. This package makes easy to use LabVIEW Vis for interacting with other embedded platforms such as Arduino, chipKIT and myRIO. With this installed in LabVIEW can easily access the device's digital inputs and outputs, analog inputs and outputs, SPI, I2C, UART, PWM and more features.

It is thanks to this package that in this project Arduino is going to be used, without writing Arduino code, making it much easier to program, thanks to visual interface offered by LabVIEW.



2.4 USING ARDUINO THROUGH LABVIEW

Once it is explained how Arduino and LabVIEW work, it is time to see how Arduino software is going to be used by using LabVIEW with the help of LINX package.

Next step is to load LINX into Arduino from LabVIEW tools. This will only need to be done once; you don't need to upload this into Arduino board every time you want to use the different software together.

It is very easy, just need to choose the device family, the device type and the firmware program interface as shown in the image below:

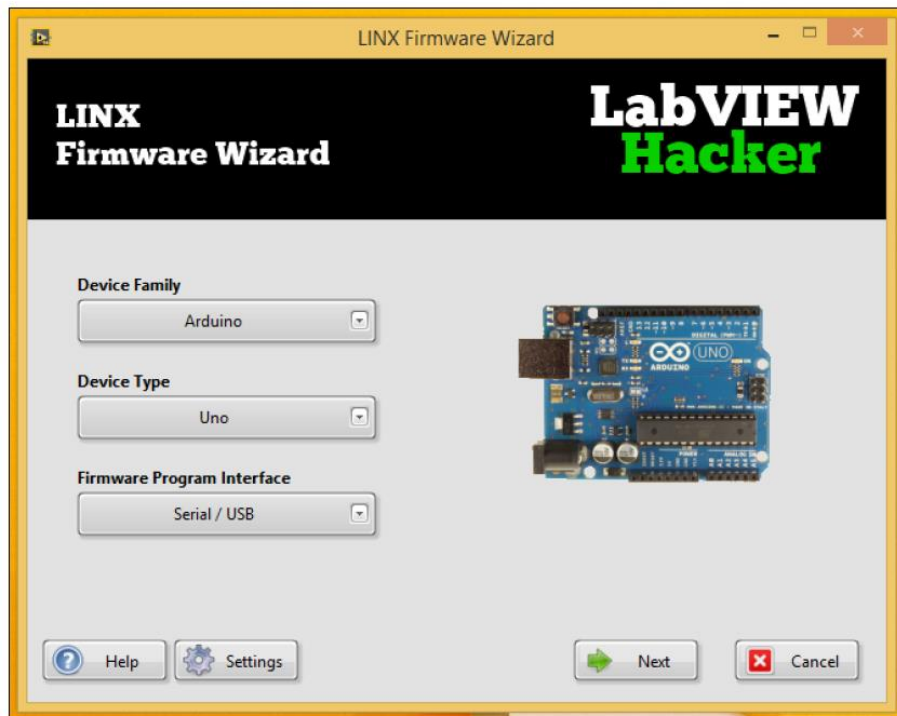


Figure 16. Interface to finally connect Arduino and LabVIEW

When everything is set up, LabVIEW software will have the LINX package installed; Arduino is ready to work by using LabVIEW programming. In the functions at the Block Diagram, there are useful functions to work with Arduino.

Functions → Makerhub → LINX

Inside LINX functions, there are several options for the user to choose.

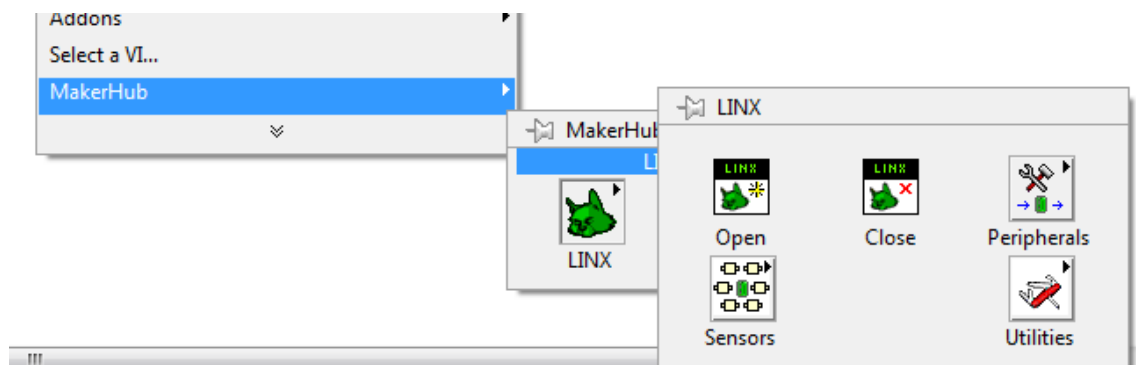
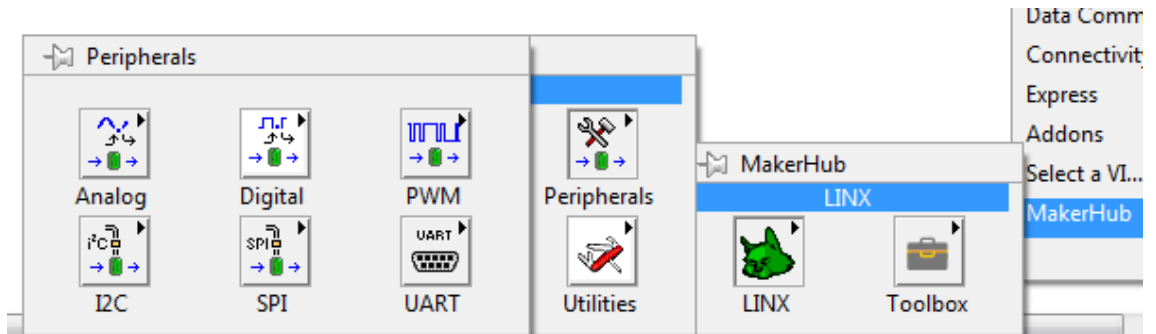


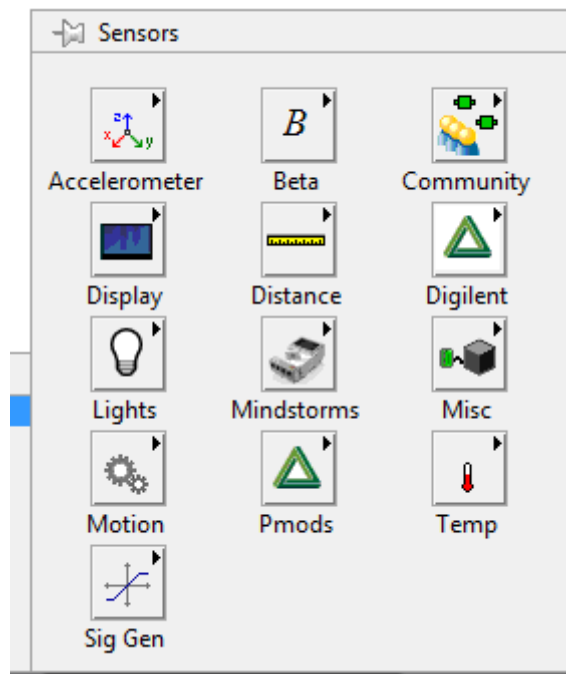
Figure 17. New LINX functions

The Open and Close functions that can be seen in the previous image are used to start and end the communication with the Arduino microcontroller.

In **Peripherals** the user finds different options for general digital and analog inputs or outputs. There are also different options for PWM, I2C, SPI and UART communication.



The user can also find **sensors** functions that are already prepared to work with specific sensors. There are or different types (as can be seen in the picture below) such as temperature sensors, light sensors, motion sensors, etc.



These functions that have been seen are really helpful when measuring parameters with different sensors. LabVIEW is very comfortable to use as the Arduino does not need to be programmed. It is also very easy to control different devices as for example a LED or electric motors thanks to the PWM function.

In conclusion, LINX package along with LabVIEW software are going to be used in order to measure different parameters as it might be temperature, and even to control a device as could be a laser beam.

3. Sensor programming

This chapter is going to give the details about which sensors have been used in the development of the project, the characteristics and specifications of the sensors, and probably what is most important of all, the programming on LabVIEW to measure different parameters with the help of these sensors.

But before describing each of the used sensors, a question must be answered, *what is a sensor?*

A sensor is a device able to detect physical or chemical magnitudes, called instrumentation variables, and to transform them into electrical parameters. These instrumentation variables can be for example intensity, temperature, light, distance, acceleration, pressure, torque, humidity, movement, pH, etc. An electric parameter might be a resistance (like a RTD), a capacitor (like a humidity sensor), an electric voltage (thermocouple), an electric current (phototransistor), etc.

The sensors are always in touch with the parameter they are measuring; therefore they are devices that use their properties in order to adapt a signal into a different type that can be processed by other devices.

The applications sensors can have are uncountable: car industry, robotics, aerospace industry, Medicine, manufacturing industries...

Moreover, in a world where automatization processes are more and more important, sensors are the devices suitable for this kind of development. It is important for an engineer to know how to use sensors and integrate them into systems, improving people's lives.



Figure 18. Group of sensors

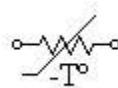
3.1 TEMPERATURE SENSOR KY-013.

The first sensor that is going to be included in this project is a temperature sensor. These kinds of sensors are one of the most used ones, as can be found almost everywhere.

The sensor chosen to work with is the **KY-013**. This sensor is a NTC thermistor. Before entering in details regarding programming with the sensor, it is important to know the kind of sensor that is going to be used.

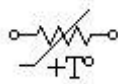
A thermistor is a temperature resistive sensor. Its functioning is based on the variation of the resistivity that a semiconductor presents against changes of temperature. The name of “thermistor” comes from **Thermally Sensitive Resistor**:

- **NTC**: Negative Temperature Coefficient. When the temperature rises, the resistance that the sensor presents decreases.



Symbol of NTC

- **PTC**: Positive Temperature Coefficient. When the temperature rises, the resistance that the sensor presents increases.



Symbol of PTC

The resistance variation of these sensors is big for small changes in temperature, but unfortunately, changes are strongly nonlinear.

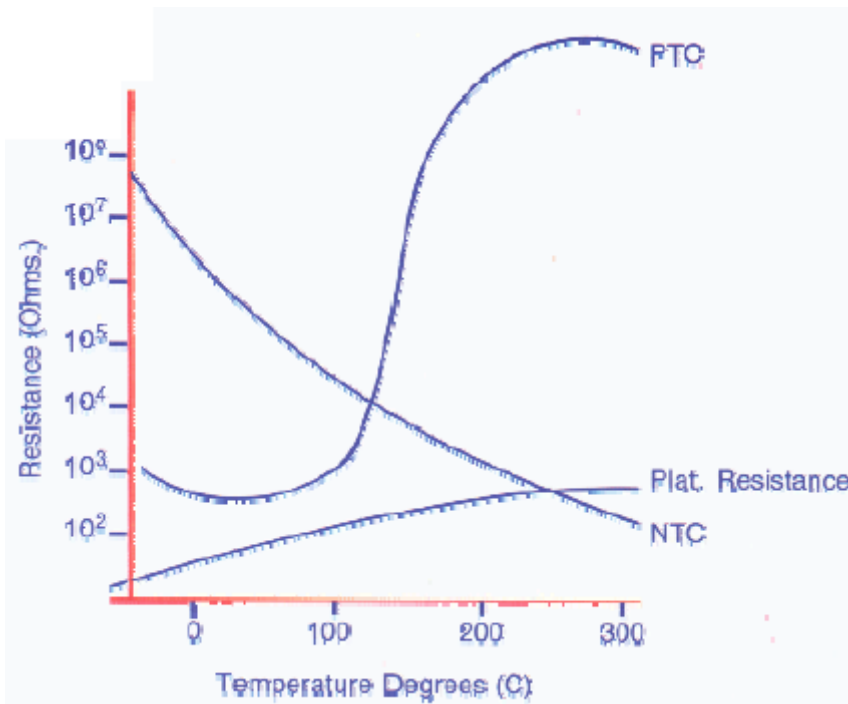


Figure 19. Resistance variations with the Temperature

As it can be seen in the previous image, PTC thermistors have a curve not suitable for measuring temperature. Therefore, the NTC thermistors are the ones that are used as temperature sensors. However, the nonlinearity makes it only possible to use NTC thermistors in short ranges of temperature, and with approximations.

- **KY-013:** this sensor includes a board that has a thermistor and a resistor mounted on it. The main specifications are: [5]

Type of sensor	Temperature Range	Accuracy
NTC Thermistor	-55°C / +125°C	+/- 0.5°C

It is important to notice that it is a 10kΩ thermistor.

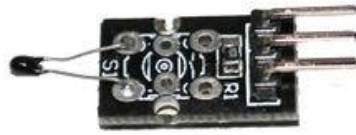


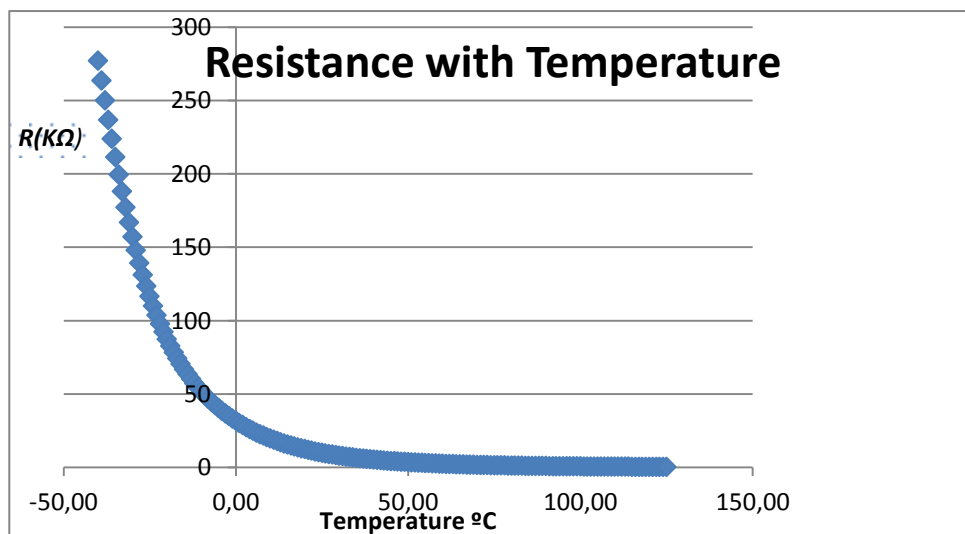
Figure 20. Thermistor NTC KY-013

As it can be seen in the image above, the sensor has three pins:

- GND pin: is connected to Arduino's GND.
- V_{cc} pin: it is connected to the supply voltage provided by Arduino. In this case 5V will be used.
- Signal pin: it is the pin that provides the voltage related to the temperature it is sensing. It **must be connected to an analog input of Arduino**.

Now that the specifications are presented, and the connections of the pins to the Arduino are clear, the next step and the most important one is to translate the voltage given by the sensor into temperature.

The first thing used for that is a table of data that relates temperature with resistance. This document can be found in the **Appendix I**. The graphic obtained from that data is the next one:



As it can be seen in the previous graph, the relationship between resistance and temperature is everything but linear. However, if you just choose a small range of

temperatures; it may be possible to approximate the function to a line, obtaining a linear relationship between the variables.

There is also another option to calculate the temperature from the voltage read by the Arduino. This is by using the **Steinhart-Hart equation**. It is not as exact as using the data from the thermistor but it gives very good results around the temperatures that this thermistor is going to be used.

The process of calculating the temperature is the following: [7]

The voltage range of this sensor, as it is going to be supplied with 5V, is the same value. The precision of the Arduino ADC converter is of 10 bit, which means there are 1024 possible states (ADC units). Therefore:

$$1024\text{ADCunits} / 5\text{V} = 204.8 \text{ ADCunits/V} \quad (1)$$

Once this is done, we have the number of ADC units the Arduino converter is managing. This value is wanted, as the resistance offered by the thermistor is calculated by using next formula:

$$R = 10\text{K} / (1023/\text{ADC} - 1) \quad (2)$$

When the resistance is calculated, the Steinhart-Hart equation is used to obtain the temperature. The original formula is the following:

$$\frac{1}{T} = A + B \ln(R) + C(\ln(R))^3 \quad (3)$$

However, this equation is hard to solve and parameters that are unknown for this thermistor. There is a simplified version and much easier to use which is:

$$\frac{1}{T} = \frac{1}{T_0} + \frac{1}{B} \ln\left(\frac{R}{R_0}\right) \quad (4)$$

In this equation we know all the parameters and can easily obtain the temperature. T_0 is room temperature, 298.15K and B is the coefficient of the thermistor. In this case $B= 3950$. R_0 is the resistance at room temperature which is 10Kohms. So finally, you just need to solve for Temperature in this equation.

Now these ideas have to be programmed with LabVIEW software so that finally the temperature is obtained and displayed on the computer.

PROGRAMMING AT LABVIEW

The following image shows how by programming at LabVIEW the temperature is measured:

Figure 21. First part of the Block Diagram.

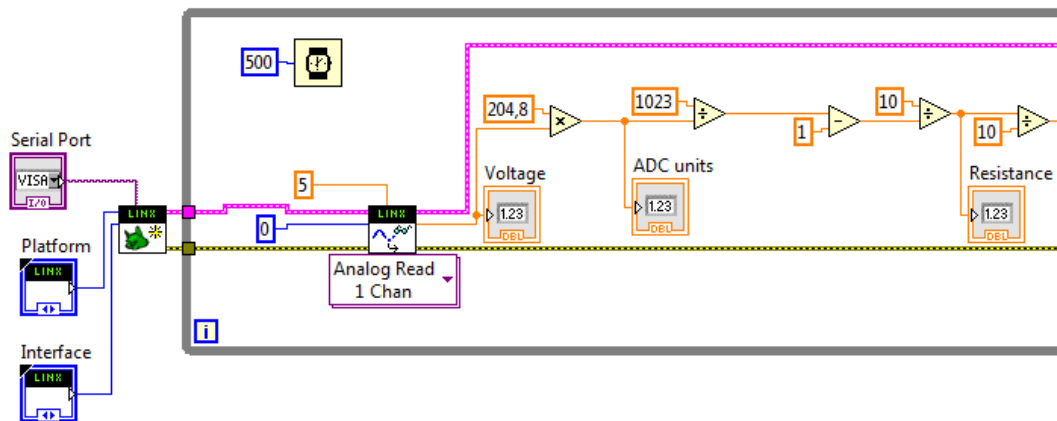
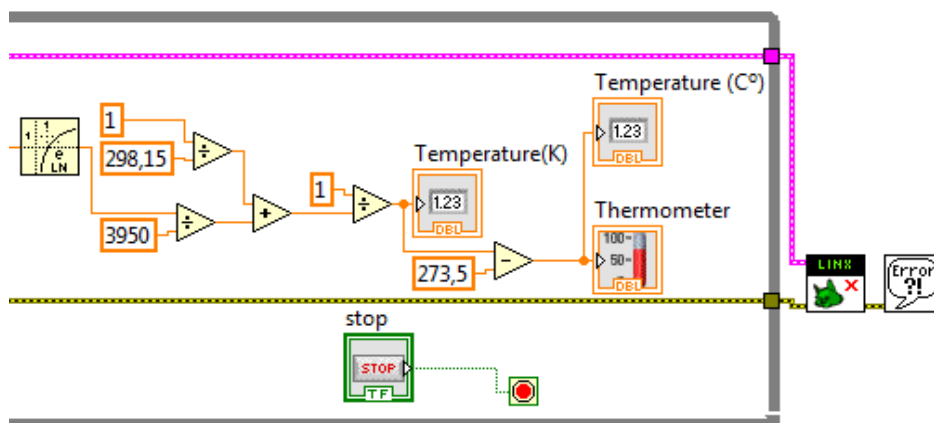
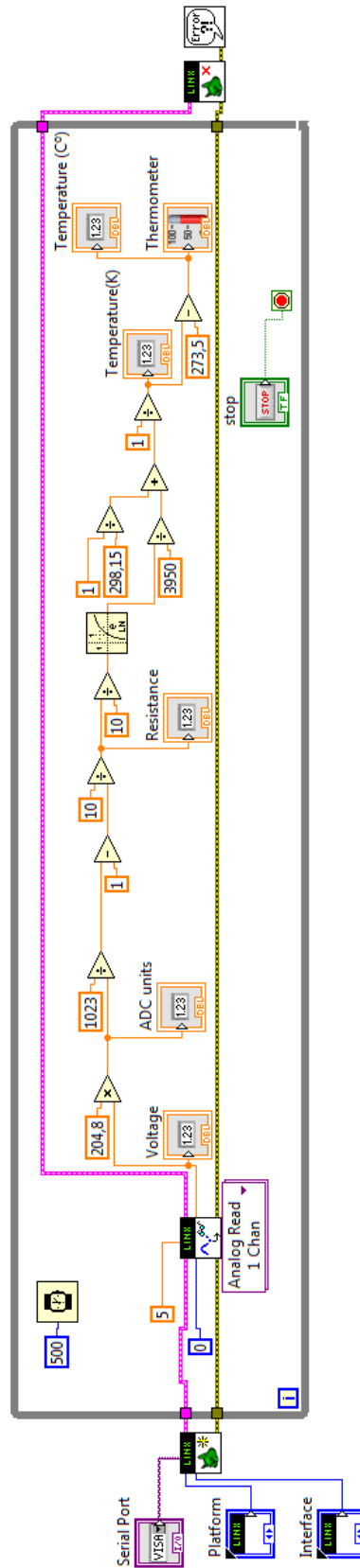


Figure 22. Second part of the Block Diagram





As it can be seen in the previous images, most of the functions are enclosed into a *WHILE* loop and with a delay of 500 ms. The selection of the PORT for the USB communication remains out of the loop.

There are two functions to start and end the communication with the Arduino that are out of the loop too. Inside the loop there is the function that allows reading the analog signal coming from the thermistor and setting the voltage reference with 5V. The signal coming out from this function is then treated as explained before so that at the end, the temperature will be showed to the user.

But the user in theory will not have access to this **Block Diagram**; what will be presented is the Front Panel and it looks like the following image:

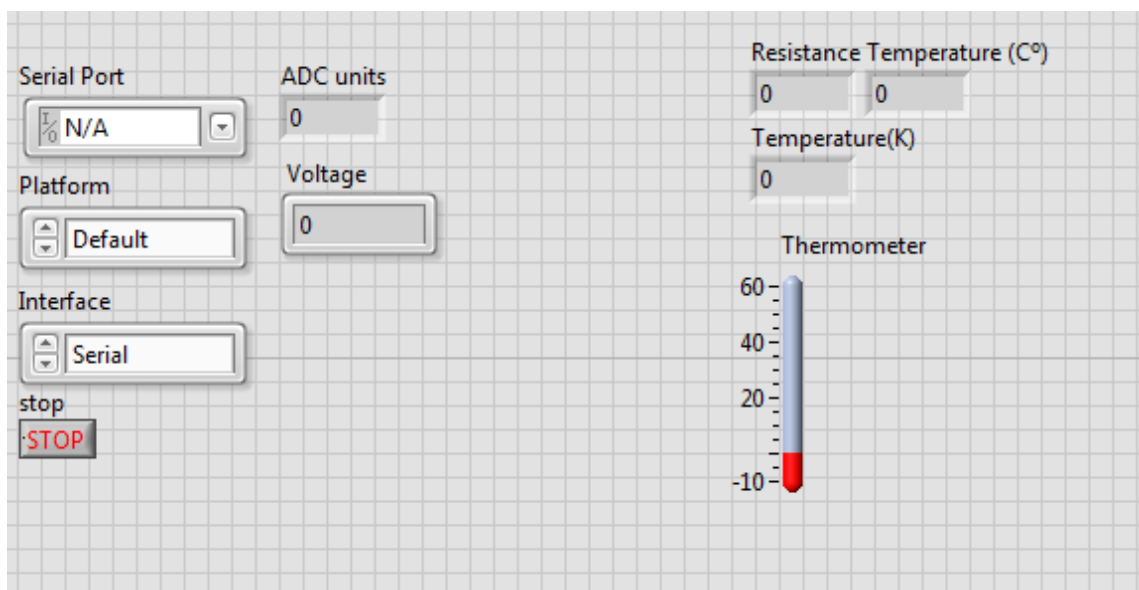


Figure 23. Front Panel

The user has to select the PORT where the Arduino is connected. Once the program is running, data will start flowing and the temperature will be shown on the temperature indicator. The thermometer diagram will show the temperature. For more information, the user is provided with the value of the resistance of the thermistor, the AD converter unit value, and the voltage of the signal.

The only control the user has in this program is the stop button that stops the execution of the program.

3.2 LIGHT SENSOR. PHOTOTRANSISTOR TEMT6000

The next parameter to be measured is the light illuminance. The **illuminance** is the total luminous flux incident on a surface, per unit area. Illuminance gives an idea of how much a place is illuminated. The unit in the international system is the **lux**.

In order to measure illuminance, a phototransistor is going to be used. A **phototransistor** is a light-sensitive transistor. The transistor is embedded into a transparent cage and photons coming from light activate the transistor. Electrons are generated by these photons in the base-collector junction and the current is injected into the base. This base current is then amplified as in a normal transistor. Thanks to the current given by the transistor, it can be known the light intensity that is hitting on the phototransistor. Therefore, if the intensity provided by the phototransistor is calculated, the illuminance can be calculated.

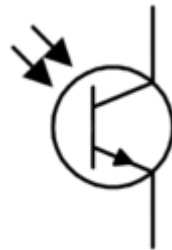


Figure 24. Symbol for a phototransistor

For the illuminance determination, a phototransistor TEMT6000 is going to be used. In order to work with it, the datasheet of the sensor has been used. Every specification is over there. **The TEMT6000 datasheet is included in the Attached File II [9].**

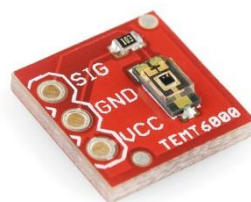


Figure 25. TEMT6000

The sensor has 3 pins that must be connected:

- **VCC**: is the voltage supply. In the project 5V are used.
- **GND**: grounding pin that is connected to Arduino's GND.
- **SIG**: the voltage given by the sensor. This pin is connected to an analog input of Arduino.

Now that the specifications are presented, and the connections of the pins to the Arduino are clear, the next step and the most important one is to translate the voltage given by the sensor into illuminance.

To be able to obtain the illuminance, it is important to know that the phototransistor has a series resistance of 10kΩ. Having the voltage and the resistance, allows obtaining the current given by the phototransistor according to Ohm's Law.

$$I = V/R \text{ (current)}$$

$$V = IR \text{ (voltage)}$$

$$R = V/I \text{ (resistance)} \text{ Ohm's Law}$$

After having the current, next step is to relate it to the illuminance. For that purpose, a graphic given in the datasheet is going to be used.

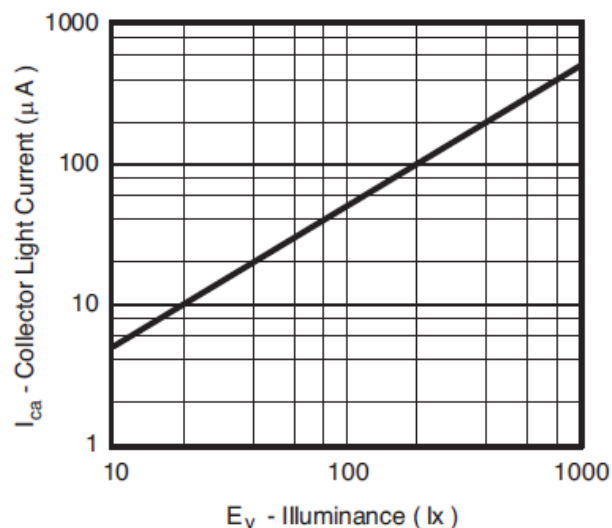


Figure 26. Illuminance relationship with collector current

Having a look at the graphic, it can be deduced that:

$$I = 1/2 \cdot \text{Illuminance} \quad (5)$$

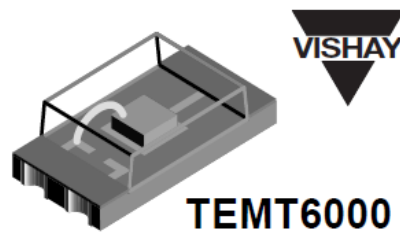
Therefore, according to this, 1 lux is equal to 2 microamperes. And this is the relationship to use in the programming at LabVIEW.

So in conclusion:

Signal voltage $\times \frac{1}{10000} \rightarrow$ Amperes \rightarrow Amperes $\times 1000000 \rightarrow$
microamperes \rightarrow microamperes/2 \rightarrow lux

$$V \times \frac{1}{10000\Omega} \times \frac{1000000\mu A}{1 A} \times \frac{1 \text{ lux}}{2\mu A} = 50 \quad (6)$$

So basically, if the signal is multiplied by 50, you get an approximation of the illuminance there is in the area around the sensor.



PROGRAMMING AT LABVIEW

The following image shows how by programming at LabVIEW the illuminance is measured:

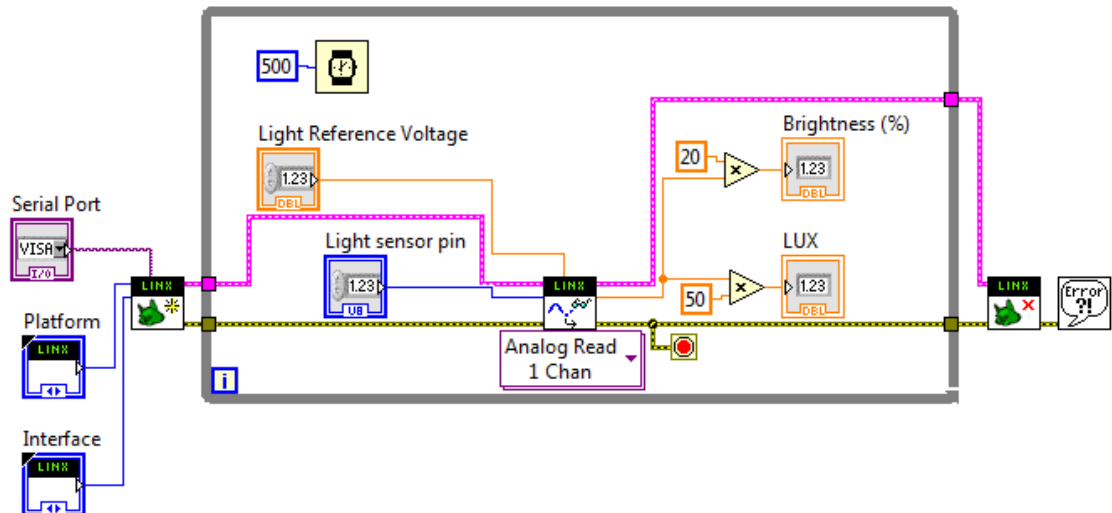


Figure 27. Block Diagram

As it can be seen in the previous image, most of the functions are enclosed into a *WHILE* loop and with a delay of 500 ms. The selection of the PORT for the USB communication remains out of the loop.

There are two functions to start and end the communication with the Arduino that are out of the loop too. Inside the loop there is the function that allows the analog signal coming from the thermistor and setting the voltage reference with a controller. The signal coming out from this function is then treated as explained before so that at the end, the illuminance will be showed to the user.

But the user in theory will not have access to this **Block Diagram**; what will be presented is the Front Panel and it looks like the following image:

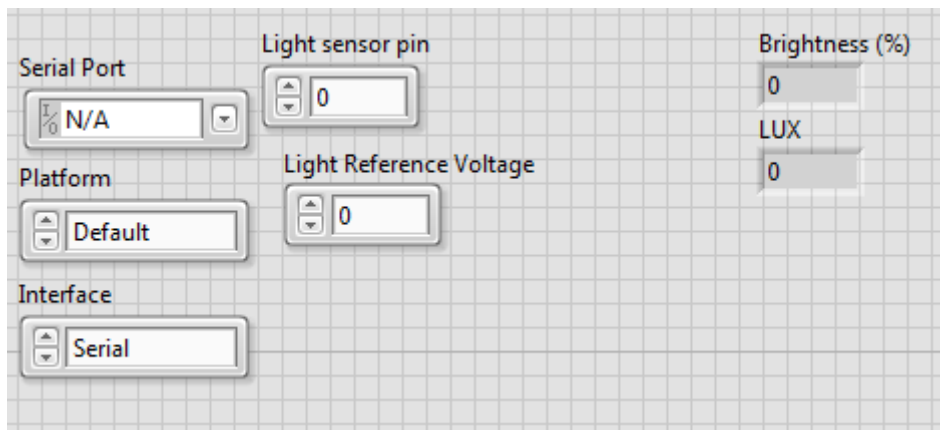


Figure 28. Front Panel

The user has to select the PORT where the Arduino is connected. In this case to give more freedom to the user, he can choose in which analog pin he wants to connect the sensor. With the controller the pin is chosen. The user has also to select the voltage supplying the sensor (in this case 5V).

Once the program is running, data will start flowing and the illuminance will be shown on the illuminance indicator. For more information, the user is provided with the value of the brightness in a scale from 0% to 100% depending on the voltage.

3.3 LASER DIODE MODULE KY-008

Next device to be used with Arduino is not going to be a sensor. In this case, the device used will be a laser diode module which will be controlled by the user from the computer as he wants.

A **laser** (from **L**ight **A**mplification by **S**timulated **E**mission of **R**adiation) is a device which uses a quantum mechanics effect, the stimulated emission, to generate a light beam. But this device emits light coherently (spatial and temporal coherence). Spatial coherence means laser can emit light into a specific spot: temporal coherence means a laser emits in a single color of light, with a specific wavelength. [3]

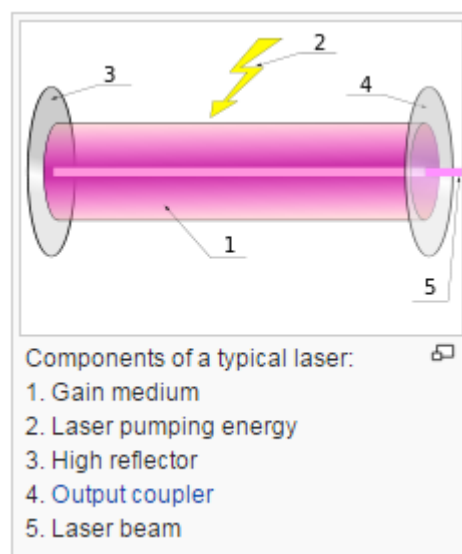


Figure 29. Laser parts

Light emissions can be so strong that lasers can be used to cut materials or even evaporate them.

Since lasers were invented in 1960, they have been used in multiple fields: Medicine, Telecommunication, manufacturing industries, war industry, architecture, commercial products, etc.

The laser used in this project is the **KY-008**. It is one of the most common devices in Arduino sensor kits. By itself, the laser is just a light emitter; however, if it is used with a laser detector, it can be used to create laser alarms for example.

It is very easy to use as it is directly powered from Arduino's pin, meaning it does not need a high power in order to work.



Figure 30. Laser diode module KY-008

In the following table, laser specifications are shown: [10]

Parameter	Value
Supply Voltage	5 V
Current	30 mA
Wavelength	650 nm
Color	Red

The connection of the pins is as shown in the image below:

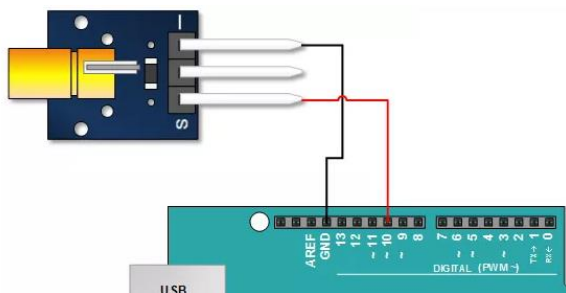


Figure 31. Connection with Arduino

As it can be seen, the S (signal) pin, is connected to a digital output. It can be any digital output, as far as the output number is well configured in LabVIEW. The (-) pin is connected to Arduino GND.

The middle pin does not need to be connected. However, this pin gives exactly the same voltage than the laser is supplied with. This pin may be used to control that the supply voltage is correct. In case the supply voltage is too high or too low, the user may be warned.

PROGRAMMING AT LABVIEW

The following image shows how by programming at LabVIEW a very simple program the laser is controlled by the user:

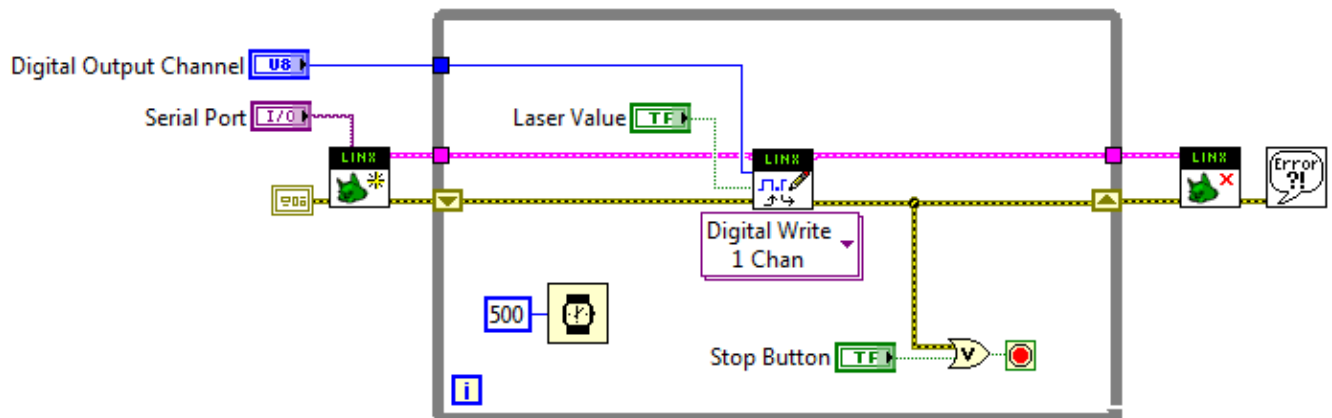


Figure 32. Block Diagram

As it can be seen in the previous image, most of the functions are enclosed into a *WHILE* loop and with a delay of 500 ms. The selection of the PORT for the USB communication remains out of the loop.

There are two functions to start and end the communication with the Arduino that are out of the loop too. Inside the loop there is the function that allows writing into a digital output (HIGH or LOW) which will turn on and off the laser.

The controlling of the output value is done with a Boolean controller that will show up at the **Front Panel** which looks like in the following image.

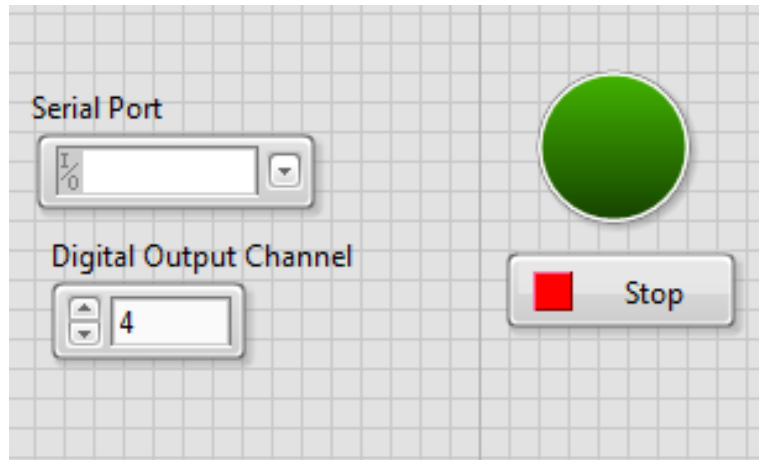


Figure 33. Front Panel

The user has to select the PORT where the Arduino is connected. In this case to give more freedom to the user, he can choose in which output pin he wants to connect the laser with the option *Digital Output Channel*.

The green button is the controller of the laser. When the user clicks on it the laser turns on and off all the times the user wants. There is also the Stop Button that will stop the running of the program if the user clicks on it.

3.4 PIR SENSOR HC-SR501

In this case, the parameter to be detected is movement. For that purpose, it is going to be used a **passive infrared sensor PIR**.

A PIR is a sensor which is able to measure the infrared light that is irradiated by the objects surrounding this sensor. This kind of electronic devices are mainly used as movement detectors.

Every single object with a temperature above absolute zero emits heat. In general, this emitted radiation is invisible for the human eye, but can be detected with the help of specific devices.

The word *passive*, used to call these sensors, refers to the fact that the devices do not emit or generate any kind of radiation in order to detect movement. [3]

The sensor used in the project is the **HC-SR501**. It is based on a pyroelectric sensor divided into two sensitive zones. These two sensitive zones, in normal conditions, are activated on the same way by the infrared radiation around the sensor. However, they emit a signal when one of these zones finds a difference with the other (as if for example it detects a person). The detection is based in the heat that emits the human body, in form of infrared radiation. The range of detection is also increased thanks to the lens above the sensor.



Figure 34.HC-SR501 sensor

For the correct movement detection, the sensor must be configured. In order to work with it, the datasheet of the device has been used. Every specification is over there. **The HC-SR501 datasheet is included in the *Attached File III [12]*.**

The sensor has 3 pins that must be connected to Arduino:

- **VCC:** is the voltage supply. In the project 5V are used.
- **GND:** grounding pin that is connected to Arduino's GND.
- **SIG:** the signal given by the sensor. In this case is a TRUE/ FALSE output. This pin is connected to a digital input of Arduino. The signal is 1 or TRUE when the sensor detects movement and is a 0 or FALSE when it does not detect anything.

Now that the specifications are presented, and the connections of the pins to the Arduino are clear, the next step is to configure the sensor. This device has 2 potentiometers and a jumper that allows the user to modify its behavior and adapt it to the user needs: precision of detection, time for reactivation, behavior facing repetitive detections...

These adjustment elements are the following:

1. **Mode selector:** the jumper that allows changing the functioning mode between continuous mode and the repetition mode. In **continuous mode**, if the sensor detects movement continuously, it will keep a continuous signal. In the **repetition mode**, the sensor will be activated when movement is detected and then will go back to its normal state; if it detects movement again, it will be activated and complete another activation cycle, but will not work on a continuous way.
2. **Sensitivity:** it increases or decreases the sensor sensitivity. The distance or the movement quantity that are needed to activate the sensor can be adjusted with this potentiometer.
3. **Timer:** this potentiometer increases or decreases the time that the sensor will be activated once it detects movement. The range goes approximately from 3 seconds up to 5 minutes.

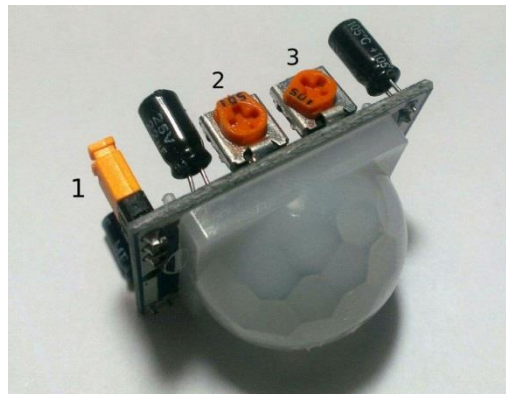


Figure 35. Potentiometers and jumper for adjusting parameters

For this project, the distance and the timer have been adjusted to their minimum. The mode used in this case is the repetition mode.

It is important to notice that after the sensor is connected, the user will have to wait for some minutes until the sensor is fully operational. This is because the sensor will be adjusting by itself and needs some time.

PROGRAMMING AT LABVIEW

The following image shows how by programming at LabVIEW the movement is detected:

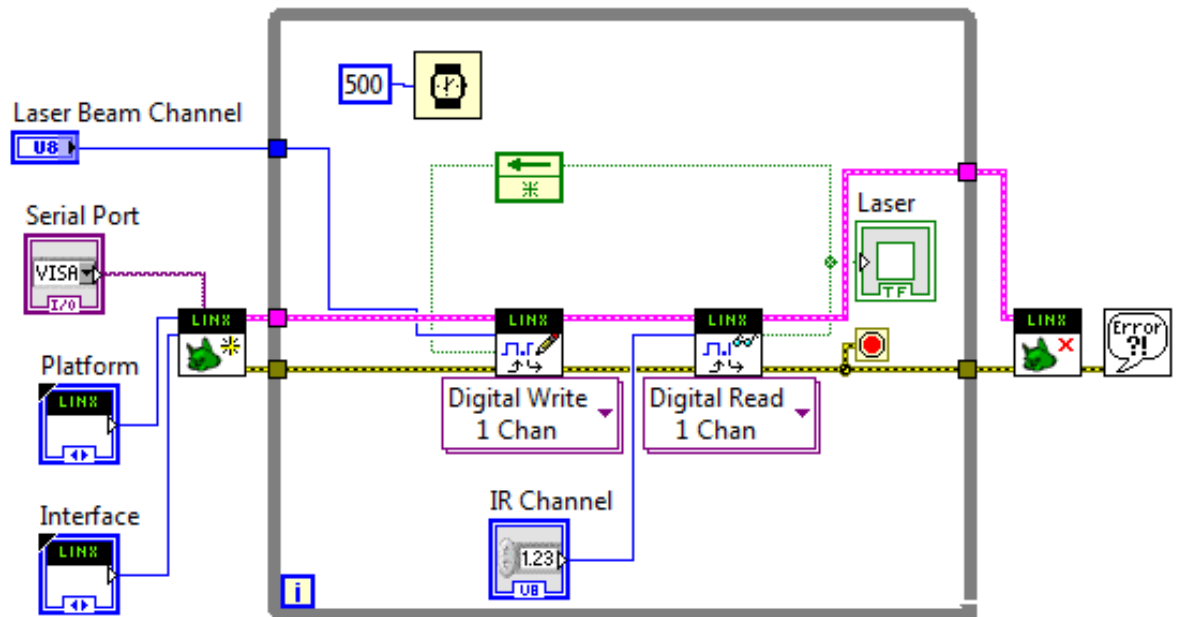


Figure 36. Block Diagram

As it can be seen in the previous image, most of the functions are enclosed into a *WHILE* loop and with a delay of 500 ms. The selection of the PORT for the USB communication remains out of the loop.

There are two functions to start and end the communication with the Arduino that are out of the loop too. Inside the loop there is the function that allows reading the digital signal coming from the PIR sensor. The signal coming from this function is Boolean: TRUE when the sensor detects movement or FALSE when there is not movement.

When there is movement detection, the TRUE value coming out from the *Digital Read Function* will enter the *Digital Write Function* and as it has been programmed, the laser will turn on. There is also an indicator that will appear on the screen showing if the laser is on and if there has been movement detection.

It is basically a modification of the laser program, in which the activation of the laser is not control by the user, but by the movement detection.

But the user in theory will not have access to this Block Diagram; what will be presented is the **Front Panel** and it looks like the following image:

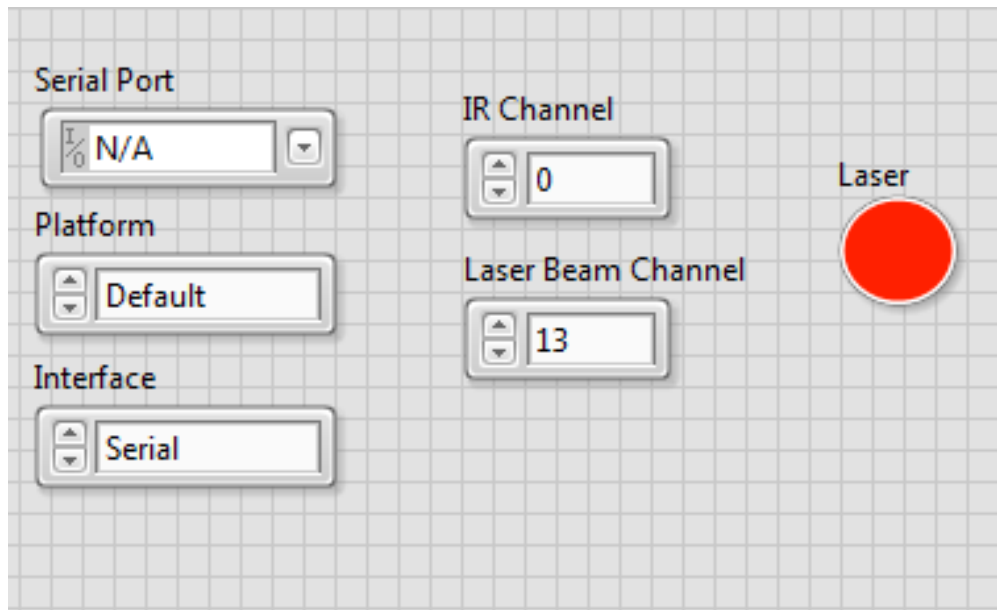


Figure 37. Front Panel

The user has to select the PORT where the Arduino is connected. Once the program is running, data will start flowing and the PIR sensor will be working and detecting movement.

The time when there is some movement detection, the laser will turn on, and the Laser indicator on the screen will change color, indicating the user about the existence of movement.

The only control the user has in this program is selection of the digital pins where the laser and the PIR sensor are connected.

3.5 MAGNETIC HALL SENSOR KY-024

A hall sensor uses the **Hall Effect** in order to measure magnetic fields or currents, and even in position determination.

If there is current flowing through the sensor and it gets near a magnetic field that flows on a vertical direction to the sensor, then the device creates a voltage proportional to the multiplication of the strength of the magnetic field and the current.

If the value of the current is well known, then the strength of the magnetic field can be measured; if a magnetic field is produced by an electric current that flows through a conductor, then that current can be measured.

In case both the magnetic field and the current that flows through a conductor are known, then the Hall sensor can be used as a metal detector.

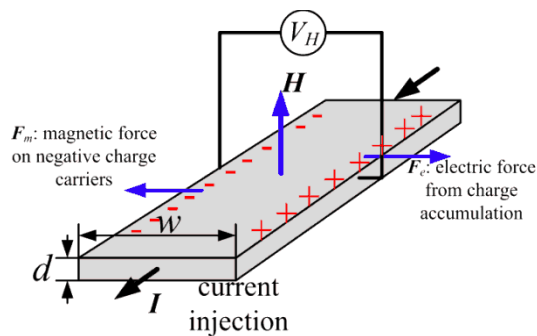


Figure 38.Hall Effect

The sensor used in the project is the **KY-024**, a linear magnetic module that has a hall sensor on it. This kind of sensor is able to detect the presence of a magnetic field and give a proportional voltage to the field's intensity and direction. [13]



Figure 39.KY-024

The KY-024 linear Hall sensor has 4 pins to be connected:

- **VCC:** is the voltage supply. In the project 5V are used.
- **GND:** grounding pin that is connected to Arduino's GND.
- **DO:** the digital output of the sensor. In case a magnetic field is detected, the pin will give 5V, a HIGH level signal.
- **AO:** it is the analog output of the sensor. The pin gives an output between 0 and 5V depending on the intensity of the magnetic field is measuring.

It is important to notice that the precision of this sensor in order to measure analog readings is not very good. That is why the sensor is going to be used in the digital way, with HIGH or LOW levels depending on if a magnetic field is detected or not. Therefore, a very simple program is going to be used so that the user can detect when there is a quite strong magnetic field near the sensor.

PROGRAMMING AT LABVIEW

The following image shows how by programming at LabVIEW the magnetic field is detected:

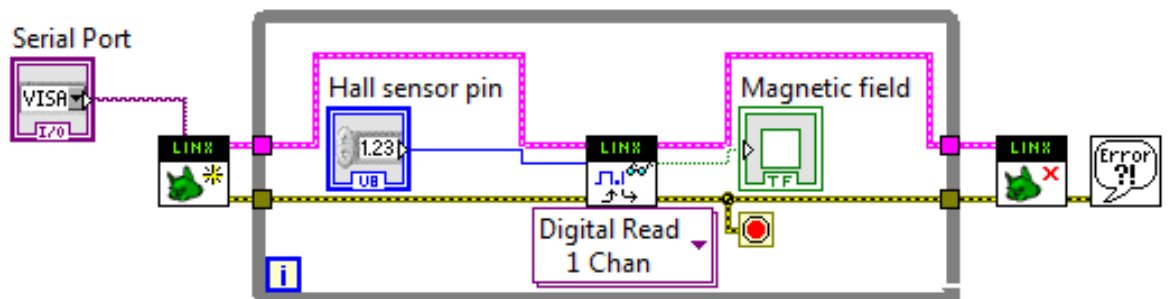


Figure 40. Block Diagram

As it can be seen in the previous image, most of the functions are enclosed into a *WHILE* loop and with a delay of 500 ms. The selection of the PORT for the USB communication remains out of the loop.

There are two functions to start and end the communication with the Arduino that are out of the loop too. Inside the loop there is the function that allows reading the digital signal coming from the Hall sensor. The signal coming from this function is Boolean: TRUE when the sensor detects a magnetic field or FALSE when there is no such magnetic field.

When there is magnetic field detection, the TRUE value coming out from the *Digital Read Function* will be seen by the user thanks to the indicator.

But the user in theory will not have access to this Block Diagram; what will be presented is the **Front Panel** and it looks like the following image:

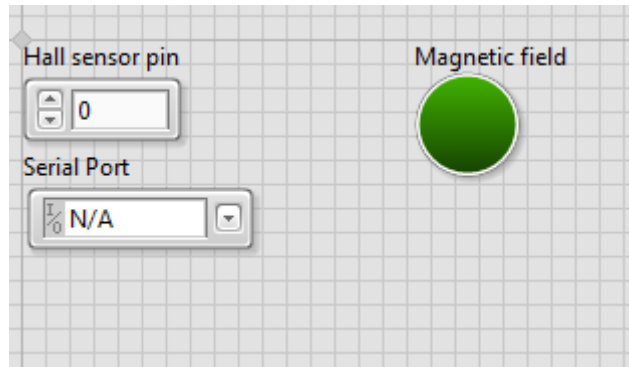


Figure 41. Front Panel

The user has to select the PORT where the Arduino is connected. Once the program is running, data will start flowing and the Hall sensor will be working and detecting magnetic fields.

The time when there is some magnetic field detection, the *Magnetic field* indicator on the screen will change color, indicating the user about the existence of a magnetic field around the sensor.

The only control the user has in this program is selection of the digital pin where the sensor is connected in the Arduino board.

3.6 WIRELESS COMMUNICATION. XBEE MODULES.

The next thing to talk about is about wireless communication when using Arduino. **XBee** is one very famous platform that can be easily used in Arduino, and perfect to use when the user wants to control a process from a computer and with no cables in the middle.

XBee is a brand given to radio modules made by *Digi International*. These radio modules need very few connections between them in order to transfer information from one to another: the power, ground, data in and data out (**UART**, universal asynchronous receiver-transmitter). [14]



Figure 42. XBee producer

Nowadays, there are different models with different characteristics to adapt to different situations. The difference in quality and functions will make the difference in the price, although for normal applications the basic modules will be good enough.



Figure 43. XBee module Series 2

But in order to communicate the computer with the Arduino wirelessly by using XBee modules, an XBee Shield and an XBee Explorer will be needed in order to achieve the objective.

The XBee shield allows the Arduino board to work with a XBee module on a very easy way, and thanks to this board the Arduino can transfer information through the air to another device (in this case the computer) a lot of feet away.

XBee is used very often due to its simplicity and in the most basic operations they can be used as a **wireless serial cable**.

In this project, by using the program of the PIR sensor, the XBee modules are going to be used in order to get a free sensor that can be located anywhere and still be monitored by the computer, without the need of any cable that would complicate the installation of it. This might be useful for example to install a people's detector.

For this, the following devices will be used:

- Arduino Board.
- Wireless Proto Shield for XBee
- 2 XBee modules
- XBee explorer module
- PIR sensor
- Wires

Once the user has all the devices and thing needed as hardware, next thing is to put them together.

One of the XBee modules has to be introduced in the XBee explorer module. This module then is connected to the computer via USB connection and is going to be the one that will send information from the computer to the Arduino.

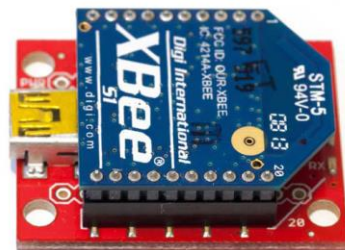


Figure 44. XBee module with in the Explorer

The other XBee module is connected into the Wireless Proto Shield like with the Explorer. Then, this Shield is mounted onto the Arduino Board. It is important to notice that with this shield, user has access to Arduino outputs and inputs. It is on this shield where the PIR sensor is connected as said before in 3.4

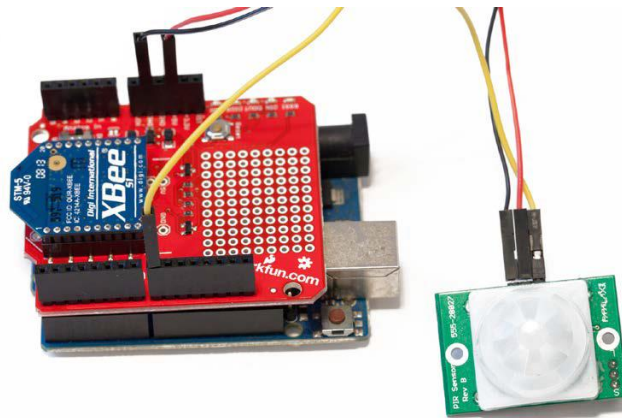


Figure 45. XBee module mounted on shield and Arduino with PIR sensor

When the hardware settings are ready, next step is to make a program that will control the sensor from the computer in LabVIEW. This is shown in the next picture:

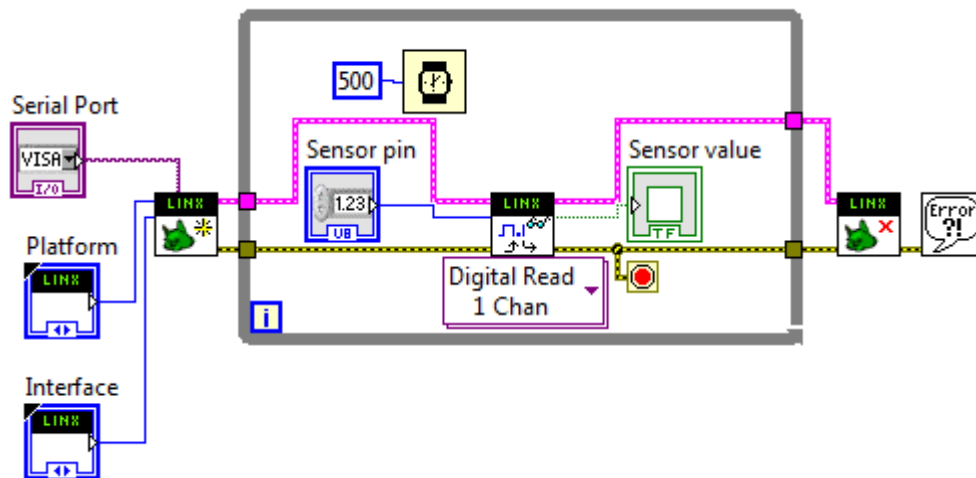


Figure 46. Block Diagram

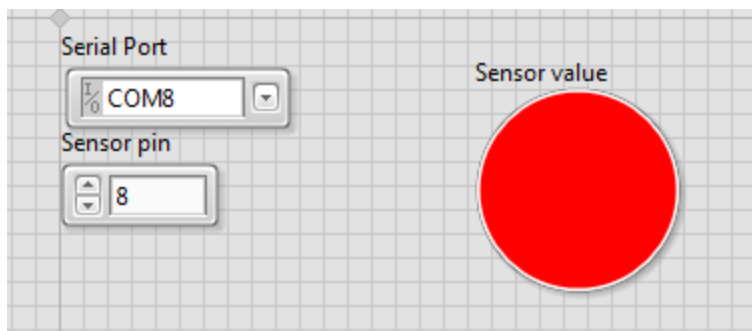


Figure 47. Front Panel

With this programming, the user is able to detect movement, and the indicator on the Front Panel changes its color. From the computer the user uploads the program to Arduino once, and configures the pin to which the sensor is connected.

Next step is to get this to work wireless. It is important to change the position of the **switch** on the Shield so that it allows wireless communication, otherwise only USB communication will work.

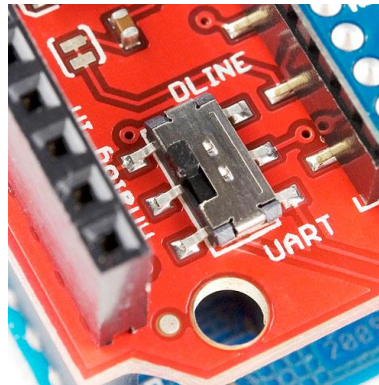


Figure 48. Switch from USB to Wireless communication

In the Front Panel, the user has to use the PORT assigned to the XBee Explorer and start the program. When this is done, the XBee modules will begin to communicate between them and the sensor signal will be controlled with the computer wirelessly. It is important that Arduino Board is power supplied, in any other case the sensor will not work.

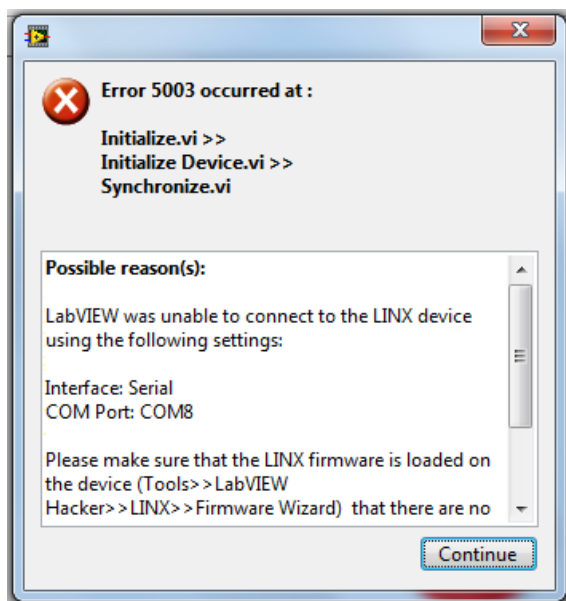


Figure 49. Error when using XBee

In the previous image, it is shown an error that appears when trying to use XBee communication. The reasons can be different and very difficult to detect.

- Firmware failure
- Drivers issue
- Incompatibilities between the module and the XBee Explorer.
- Incompatibilities between the module and the Wireless Proto Shield
- Malfunctioning of some of the two modules
- Others

It has been tried to fix the error, but it has not been possible to get the modules to work. One of the solutions could be to change them for new ones and make sure they are compatible with Arduino board, the shield and the explorer.

However, thanks to the help of Prof. Dr. G. Nikolov, it was noticed that the problem was a **firmware issue**.

Once the firmware is updated, the problem is solved and the wireless communication can be used without any trouble. The wireless alarm works just fine and can detect movement when someone passes by the sensor.

The **main advantage** is that you can place the alarm anywhere without the need of having the computer connected by cable to the Arduino UNO, giving the user a lot of freedom and a wide range of possibilities when using the devices.

Note: it is important to know that XBee wireless communication is **not suitable** to be used when there is a lot of **noise** and interferences in the environment. For example, in factories with powerful machines would not be recommendable to use this kind of modules, as the communication between them could fail due to electromagnetic noise around.

4. Sensor functioning

Once the sensors have been used in the laboratory with the programs seen in the previous chapter, it is important to make a brief explanation of these sensors' behavior.

- **KY-013 Temperature sensor:** as this temperature sensor is an NTC thermistor, the strong nonlinearity that it presents makes it really hard to obtain an accurate measure of the temperature in a wide range. However, pretty good measures have been obtained on temperatures near the environment temperature.

It is important to notice that the sensor's slew rate is very low; it needs a lot of time to stabilize when there is a sudden change on the temperature. This should be taken into account for real applications.

- **TEMT6000 Light phototransistor:** thanks to the datasheet of this sensor, it is quite easy to use it. Obtaining the approximate illuminance is simple, and with a pretty nice accuracy. For common applications it can be perfectly used. Its response is quite fast. However, if you need a sensor for a more critical application, you may need to use a different one.
- **KY-008 Laser:** this laser consumes very low rates of energy. Therefore it is perfect for systems working with batteries. However, its low power reduces the possibilities of been used in complex systems.
- **HC-SR501 PIR:** the PIR sensor used in the laboratory actually offers many possibilities. Once it is well configured, the detection is quite sensitive; therefore the accuracy is pretty good for general purposes. The response is really fast and there are no false alarms of movement.

- **KY-024:** the Hall sensor used in the laboratory has its limitations. It is not a good choice if the user wants to measure the magnetic field intensity, the error in the measure will be too high. However, this sensor offers a performance good enough to be able to detect when there is a magnetic field around the sensor and give a digital signal.

Note: the sensor range is short. It only works when the magnet is a few millimeters away from the sensor. It will not work for long distances.

5. Applications or Uses

Now that the sensors have been tried on the laboratory, it is important to think on which systems these devices may be useful in order to improve people's lives.

- **KY-013 Temperature sensor:** this sensor could be used in applications where it is not important the time of reaction. For example it could be useful in **home automation** in a system that controls a house temperature which will always be around 20°C and with a narrow range of variation.

It could be also used to sense the temperature in **gardens** or **greenhouses** for the proper growth of the plants and vegetables.



Figure 50. Digital thermometer

- **TEMT6000 Light phototransistor:** this is a perfect sensor for common applications. For example, **smart street lamps** which turn on when the light is below certain level. This could be also used in **home automation** as automatic lights controller.

More precise applications would be to use these sensors in **mobile phones**, tablets or electronical devices to control for example the screen brightness.

Furthermore, the **car industry** could also use these sensors. In the automation of headlights, TEMT6000 could be used.



Figure 51. Street lamp with sensor

- **KY-008 Laser:** the low energy consumption makes the laser suitable to use it in system where power is really important.

Combined with a laser receptor, this laser will be perfect to be used in an **alarm system**.



Figure 52. Laser Alarm

- **HC-SR501:** the PIR sensor applications are a lot. However, the sensor will most of the times used as a **movement detector**. But thanks to the different possible configurations it can be used in very different situations and places.

Movement detectors based on this PIR sensor could be used for **automatically sensing light** for a room, bathroom, basement, porch, warehouse, garage, etc. It can also be used to get **ventilators** to work. In safety applications, obviously it can work as an **alarm** to detect when a person is entering some place.



Figure 53. Movement detector

- **KY-024:** this kind of Hall sensor is mainly used as a position sensor, not for measuring field intensities. For example, it is used in the **car industry**, for example to detect if seat belts are fastened or not. They can also be used to detect the position of the pedals of vehicles or the position of seats. Even **bike speedometers** use these Hall sensors.

These sensors can be used also in integrated circuits: on **printers** to control the motor position, on computers' **HDD** (hard disk drives), **ventilators**.

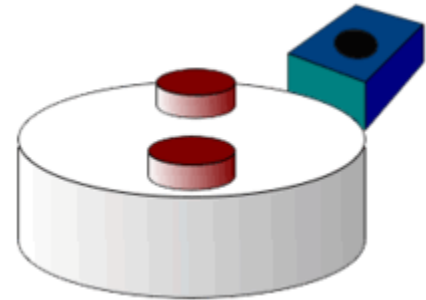
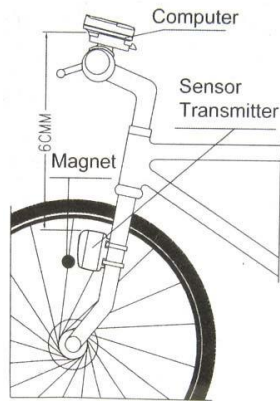


Figure 54. Position hall sensor

6. Budget

A little budget of the elements that have been used in the laboratory is presented here:

	Units	Price per unit (\$)	Total price (\$)
Arduino UNO	1	29	29
Wires (65 pcs)	1	2,4	2,4
Breadboard	1	2,8	2,8
KY-013 (temp)	1	4,79	4,79
TEMT6000 (light)	1	4,95	4,95
KY-008 (laser)	1	6,99	6,99
HC-SR501 (PIR)	1	5,99	5,99
KY-024 (Hall sensor)	1	1,99	1,99
Xbee module S1	2	24,95	49,9
Xbee Explorer	1	24,95	24,95
Wireless Proto Shield	1	19,95	19,95
		TOTAL	153,71 \$

**Prices have been taken from Amazon [4]*

7. Conclusions

LabVIEW is software which makes working with Arduino platform very comfortable, as you do not need to program with Arduino language. All the programming is visual thanks to the LabVIEW functions, which is easy and very intuitive.

There are thousands of different possibilities and applications that LabVIEW together with Arduino can be implemented. In the project, several sensors have been controlled with this software.

This has shown how measuring and controlling different parameters as light, temperature, magnetic fields, etc. can be easily carried from the Arduino microcontroller to the computer. In weather stations or in industry's processes this can be very helpful, and the worker can easily use the Front Panel of the computer.

Moreover, using wireless communication as with XBee radio modules, gives a freedom that the user may need. A possible use can be in an alarm system as have been shown with the movement detectors.

In conclusion, engineering in every field requires controlling thousands of parameters and operations; these operations are been automatized and knowledge of software that allows this automatization, is basic. In this project, using LabVIEW and Arduino gives the engineer a global vision of how automatic control and the use of different sensors work.



8. Bibliography

Books

- [1] Marco Schwartz and Oliver Manickum, “*Programming Arduino with LabVIEW*”, ISBN 978-1-84969-822-1, January 2015 Edition
- [2] Ronald W. Larsen, “*LabVIEW for engineers*”, ISBN 978-0-13-609429-6, 2011 Edition

General information

[3] *Wikipedia*

[4] *Amazon* (price information)

Temperature Sensor

[5] <https://rydepier.wordpress.com/2015/08/04/thermistor-keyes/> (29/04/2016)

[6] https://tkkrilab.nl/wiki/Arduino_KY-013_Temperature_sensor_module (29/04/2016)

[7] <https://learn.adafruit.com/thermistor/using-a-thermistor> (29/04/2016)

Light Sensor

[8] <http://forum.arduino.cc/index.php?topic=185158.0> (29/04/2016)

[9] Datasheet at Appendix II

Laser

[10] <http://henrysbench.capnfatz.com/henrys-bench/arduino-output-devices/ky-008-arduino-laser-module-guide-and-tutorial/> (30/04/2016)

PIR sensor

[11] <http://www.instructables.com/id/Ejemplo-b%C3%A1sico-con-HC-SR501/>
(02/05/2016)

[12] Datasheet at Appendix III

Hall sensor

[13] <http://greppipelinix.synology.me/?p=207> (12/05/2016)

XBee

[14] <https://learn.sparkfun.com/tutorials/xbee-shield-hookup-guide> (12/05/2016)

[15] <http://www.digi.com/> (12/05/2016)

May 18th 2016



Megara Veramendi Fernández

Appendix I:

Thermistor NTC KY-013 data

Resistance @ 25°C = 10K		B =3950K						
T (°C)	R(KΩ)	T (°C)	R(KΩ)	T (°C)	R(KΩ)	T (°C)	R(KΩ)	T (°C)
-40.00	277.2	1.00	30.25	42.00	4.915	83.00	1.128	124.00
-39.00	263.6	2.00	28.82	43.00	4.723	84.00	1.093	125.00
-38.00	250.1	3.00	27.45	44.00	4.539	85.00	1.059	126.00
-37.00	236.8	4.00	26.16	45.00	4.363	86.00	1.027	127.00
-36.00	224	5.00	24.94	46.00	4.195	87.00	0.9955	128.00
-35.00	211.5	6.00	23.77	47.00	4.034	88.00	0.9654	129.00
-34.00	199.6	7.00	22.67	48.00	3.88	89.00	0.9363	130.00
-33.00	188.1	8.00	21.62	49.00	3.733	90.00	0.9083	131.00
-32.00	177.3	9.00	20.63	50.00	3.592	91.00	0.8812	132.00
-31.00	167	10.00	19.68	51.00	3.457	92.00	0.855	133.00
-30.00	157.2	11.00	18.78	52.00	3.328	93.00	0.8297	134.00
-29.00	148.1	12.00	17.93	53.00	3.204	94.00	0.8052	135.00
-28.00	139.4	13.00	17.12	54.00	3.086	95.00	0.7816	136.00
-27.00	131.3	14.00	16.35	55.00	2.972	96.00	0.7587	137.00
-26.00	123.7	15.00	15.62	56.00	2.863	97.00	0.7366	138.00
-25.00	116.6	16.00	14.93	57.00	2.759	98.00	0.7152	139.00
-24.00	110	17.00	14.26	58.00	2.659	99.00	0.6945	140.00
-23.00	103.7	18.00	13.63	59.00	2.564	100.00	0.6744	141.00
-22.00	97.9	19.00	13.04	60.00	2.472	101.00	0.6558	142.00
-21.00	92.5	20.00	12.47	61.00	2.384	102.00	0.6376	143.00
-20.00	87.43	21.00	11.92	62.00	2.299	103.00	0.6199	144.00
-19.00	82.79	22.00	11.41	63.00	2.218	104.00	0.6026	145.00
-18.00	78.44	23.00	10.91	64.00	2.141	105.00	0.5858	146.00
-17.00	74.36	24.00	10.45	65.00	2.066	106.00	0.5694	147.00
-16.00	70.53	25.00	10	66.00	1.994	107.00	0.5535	148.00
-15.00	66.92	26.00	9.575	67.00	1.926	108.00	0.538	149.00
-14.00	63.54	27.00	9.17	68.00	1.86	109.00	0.5229	150.00
-13.00	60.34	28.00	8.784	69.00	1.796	110.00	0.5083	151.00
-12.00	57.33	29.00	8.416	70.00	1.735	111.00	0.4941	152.00
-11.00	54.5	30.00	8.064	71.00	1.677	112.00	0.4803	153.00
-10.00	51.82	31.00	7.73	72.00	1.621	113.00	0.4669	154.00

-9.00	49.28	32.00	7.41	73.00	1.567	114.00	0.4539	155.00
-8.00	46.89	33.00	7.106	74.00	1.515	115.00	0.4412	156.00
-7.00	44.62	34.00	6.815	75.00	1.465	116.00	0.429	157.00
-6.00	42.48	35.00	6.538	76.00	1.417	117.00	0.4171	158.00
-5.00	40.45	36.00	6.273	77.00	1.371	118.00	0.4055	159.00
-4.00	38.53	37.00	6.02	78.00	1.326	119.00	0.3944	160.00
-3.00	36.7	38.00	5.778	79.00	1.284	120.00	0.3835	161.00
-2.00	34.97	39.00	5.548	80.00	1.243	121.00	0.373	162.00
-1.00	33.33	40.00	5.327	81.00	1.203	122.00	0.3628	163.00
0.00	31.77	41.00	5.117	82.00	1.165	123.00	0.353	164.00

Appendix II:

Light sensor

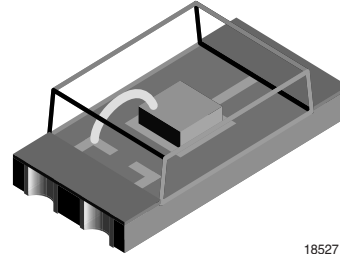
Phototransistor

TEEMT6000

Ambient Light Sensor

Description

TEMT6000 is a silicon NPN epitaxial planar phototransistor in a miniature transparent mold for surface mounting onto a printed circuit board. The device is sensitive to the visible spectrum.



18527

Features

- Adapted to human eye responsivity
- Wide angle of half sensitivity $\phi = \pm 60^\circ$
- SMD style package on PCB technology
- Suitable for IR reflow soldering
- Lead free component
- Component in accordance to RoHS 2002/95/EC and WEEE 2002/96/EC

Applications

Ambient light sensor for display backlight dimming in:

Mobile phones
 Notebook computers
 PDA's
 Cameras
 Dashboards

Absolute Maximum Ratings

$T_{amb} = 25^\circ\text{C}$, unless otherwise specified

Parameter	Test condition	Symbol	Value	Unit
Collector Emitter Voltage		V_{CEO}	6	V
Emitter Collector Voltage		V_{ECO}	1.5	V
Collector current		I_C	20	mA
Total Power Dissipation	$T_{amb} \leq 55^\circ\text{C}$	P_{tot}	100	mW
Junction Temperature		T_j	100	$^\circ\text{C}$
Operating Temperature Range		T_{amb}	- 40 to + 85	$^\circ\text{C}$
Storage Temperature Range		T_{stg}	- 40 to + 85	$^\circ\text{C}$
Soldering Temperature	$t \leq 3\text{ s}$	T_{sd}	260	$^\circ\text{C}$
Thermal Resistance Junction/ Ambient		R_{thJA}	450	K/W

Basic Characteristics

$T_{amb} = 25\text{ }^{\circ}\text{C}$, unless otherwise specified

Parameter	Test condition	Symbol	Min	Typ.	Max	Unit
Collector Emitter Breakdown Voltage	$I_C = 0.1\text{ mA}$	V_{CEO}	6			V
Collector dark current	$V_{CE} = 5\text{ V}, E = 0$	I_{CEO}		3	50	nA
Collector-emitter capacitance	$V_{CE} = 0\text{ V}, f = 1\text{ MHz}, E = 0$	C_{CEO}		16		pF
Collector Light Current	$E_V = 20\text{ lx}$, standard light A	I_{ca}	3.5	10	16	μA
	$E_V = 100\text{ lx}$, standard light A	I_{ca}		50		μA
Angle of Half Sensitivity		ϕ		± 60		deg
Wavelength of Peak Sensitivity		λ_p		570		nm
Range of Spectral Bandwidth		$\lambda_{0.1}$		360 to 970		nm
Collector Emitter Saturation Voltage	$E_V = 20\text{ lx}, 0.45\text{ }\mu\text{A}$	V_{CEsat}		0.1		V

Typical Characteristics ($T_{amb} = 25\text{ }^{\circ}\text{C}$ unless otherwise specified)

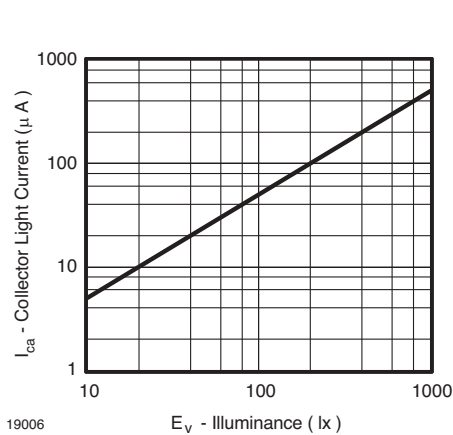


Figure 1. Collector Light Current vs. Illuminance

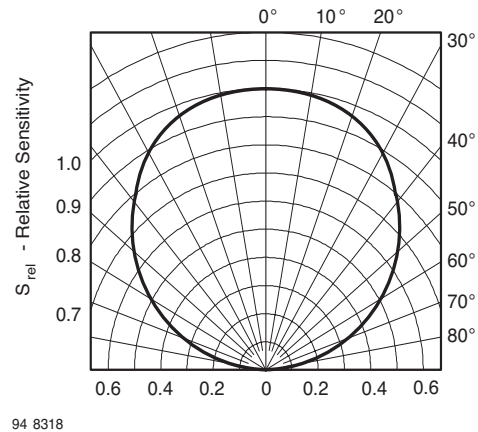


Figure 3. Relative Radiant Sensitivity vs. Angular Displacement

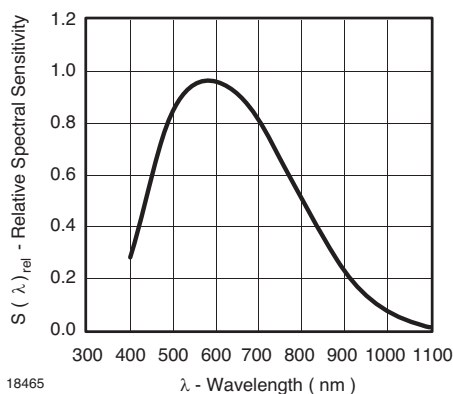


Figure 2. Relative Spectral Responsivity vs. Wavelength

Reflow Solder Profiles

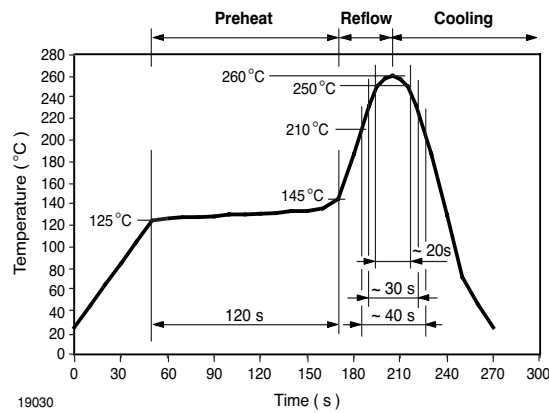


Figure 4. Lead-Free (Sn) Reflow Solder Profile

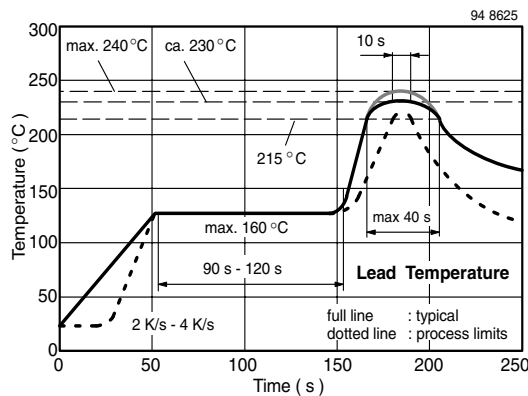


Figure 5. Lead Tin (SnPb) Reflow Solder Profile

Drying

In case of moisture absorption devices should be baked before soldering. Conditions see J-STD-020 or Label. Devices taped on reel dry using recommended conditions 192 h @ 40 °C (+ 5 °C), RH < 5 %

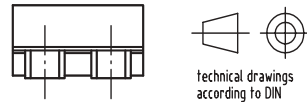
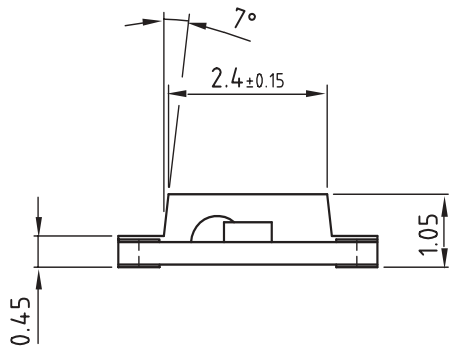
Drypack

Devices are packed in moisture barrier bags (MBB) to prevent the products from moisture absorption during transportation and storage. Each bag contains a desiccant.

Floor Life

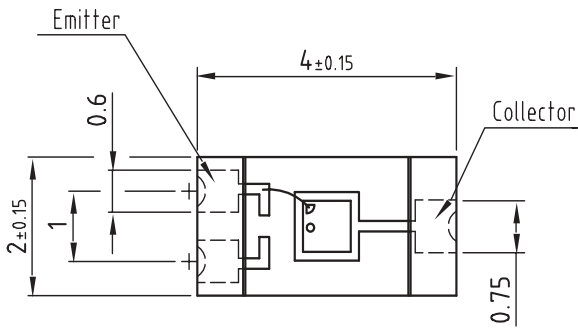
Floor life (time between soldering and removing from MBB) must not exceed the time indicated in J-STD-020. TEMT6000 is released for: Moisture Sensitivity Level 4, according to JEDEC, J-STD-020
 Floor Life: 72 h
 Conditions: $T_{amb} < 30\text{ °C}$, RH < 60 %

Package Dimensions in mm

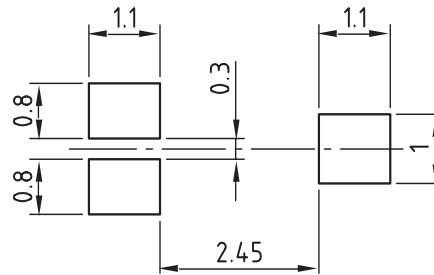


technical drawings
according to DIN
specifications

All dimensions in mm
Not indicated tolerances ± 0.1



Recommended solder pad
Footprint



Drawing-No.: 6.541-5053.01-4
Issue: 2; 21.06.04

18464

Ozone Depleting Substances Policy Statement

It is the policy of **Vishay Semiconductor GmbH** to

1. Meet all present and future national and international statutory requirements.
2. Regularly and continuously improve the performance of our products, processes, distribution and operating systems with respect to their impact on the health and safety of our employees and the public, as well as their impact on the environment.

It is particular concern to control or eliminate releases of those substances into the atmosphere which are known as ozone depleting substances (ODSs).

The Montreal Protocol (1987) and its London Amendments (1990) intend to severely restrict the use of ODSs and forbid their use within the next ten years. Various national and international initiatives are pressing for an earlier ban on these substances.

Vishay Semiconductor GmbH has been able to use its policy of continuous improvements to eliminate the use of ODSs listed in the following documents.

1. Annex A, B and list of transitional substances of the Montreal Protocol and the London Amendments respectively
2. Class I and II ozone depleting substances in the Clean Air Act Amendments of 1990 by the Environmental Protection Agency (EPA) in the USA
3. Council Decision 88/540/EEC and 91/690/EEC Annex A, B and C (transitional substances) respectively.

Vishay Semiconductor GmbH can certify that our semiconductors are not manufactured with ozone depleting substances and do not contain such substances.

**We reserve the right to make changes to improve technical design
and may do so without further notice.**

Parameters can vary in different applications. All operating parameters must be validated for each customer application by the customer. Should the buyer use Vishay Semiconductors products for any unintended or unauthorized application, the buyer shall indemnify Vishay Semiconductors against all claims, costs, damages, and expenses, arising out of, directly or indirectly, any claim of personal damage, injury or death associated with such unintended or unauthorized use.

Vishay Semiconductor GmbH, P.O.B. 3535, D-74025 Heilbronn, Germany
Telephone: 49 (0)7131 67 2831, Fax number: 49 (0)7131 67 2423

This datasheet has been download from:

www.datasheetcatalog.com

Datasheets for electronics components.

Appendix III:

PIR sensor

HC-SR501

HC-SR501 PIR MOTION DETECTOR

Product Description

HC-SR501 is based on infrared technology, automatic control module, using Germany imported LHI778 probe design, high sensitivity, high reliability, ultra-low-voltage operating mode, widely used in various auto-sensing electrical equipment, especially for battery-powered automatic controlled products.

Specification:

- Voltage: 5V – 20V
- Power Consumption: 65mA
- TTL output: 3.3V, 0V
- Delay time: Adjustable (.3->5min)
- Lock time: 0.2 sec
- Trigger methods: L – disable repeat trigger, H enable repeat trigger
- Sensing range: less than 120 degree, within 7 meters
- Temperature: – 15 ~ +70
- Dimension: 32*24 mm, distance between screw 28mm, M2, Lens dimension in diameter: 23mm

Application:

Automatically sensing light for Floor, bathroom, basement, porch, warehouse, Garage, etc, ventilator, alarm, etc.

Features:

- Automatic induction: to enter the sensing range of the output is high, the person leaves the sensing range of the automatic delay off high, output low.
- Photosensitive control (optional, not factory-set) can be set photosensitive control, day or light intensity without induction.
- Temperature compensation (optional, factory reset): In the summer when the ambient temperature rises to 30 ° C to 32 ° C, the detection distance is slightly shorter, temperature compensation can be used for performance compensation.
- Triggered in two ways: (jumper selectable)
 - non-repeatable trigger: the sensor output high, the delay time is over, the output is automatically changed from high level to low level;
 - repeatable trigger: the sensor output high, the delay period, if there is human activity in its sensing range, the output will always remain high until the people left after the delay will be high level goes low (sensor module detects a time delay period will be automatically extended every human activity, and the starting point for the delay time to the last event of the time).
- With induction blocking time (the default setting: 2.5s blocked time): sensor module after each sensor output (high into low), followed by a blockade set period of time, during this time period sensor does not accept any sensor signal. This feature can be achieved sensor output time "and" blocking time "interval between the work can be applied to interval detection products; This function can inhibit a variety of interference in the process of load switching. (This time can be set at zero seconds – a few tens of seconds).
- Wide operating voltage range: default voltage DC4.5V-20V.
- Micropower consumption: static current <50 microamps, particularly suitable for battery-powered automatic control products.
- Output high signal: easy to achieve docking with the various types of circuit.

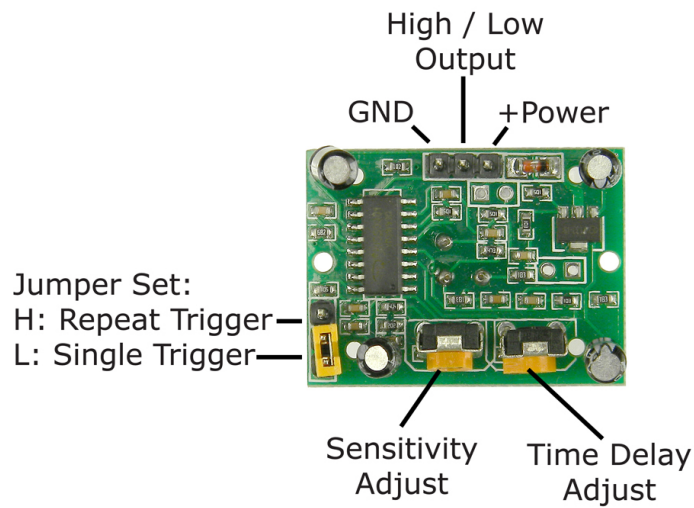
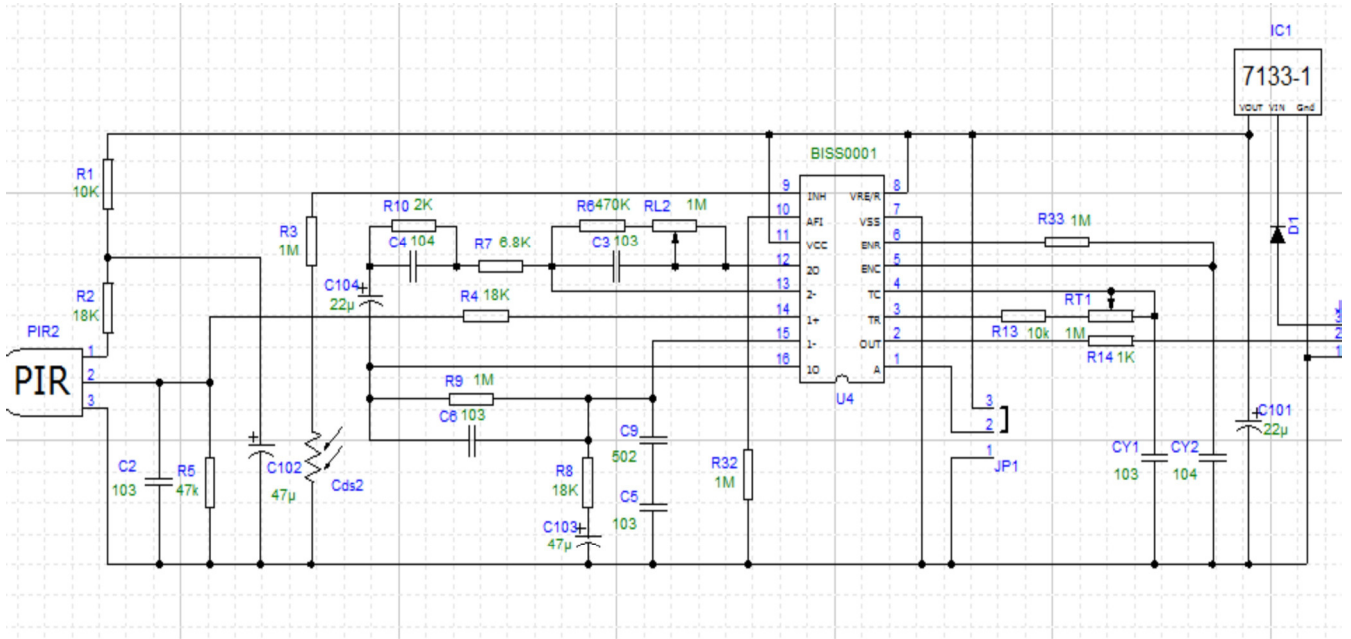
Adjustment:

- Adjust the distance potentiometer clockwise rotation, increased sensing distance (about 7 meters), on the contrary, the sensing distance decreases (about 3 meters).
- Adjust the delay potentiometer clockwise rotation sensor the delay lengthened (300S), on the contrary, shorten the induction delay (5S).

Instructions for use:

- Sensor module is powered up after a minute, in this initialization time intervals during this module will output 0-3 times, a minute later enters the standby state.
- Should try to avoid the lights and other sources of interference close direct module surface of the lens, in order to avoid the introduction of interference signal malfunction; environment should avoid the wind flow, the wind will cause interference on the sensor.
- Sensor module with dual probe, the probe window is rectangular, dual (A B) in both ends of the longitudinal direction
 - so when the human body from left to right or right to left through the infrared spectrum to reach dual time, distance difference, the greater the difference, the more sensitive the sensor,
 - when the human body from the front to the probe or from top to bottom or from bottom to top on the direction traveled, double detects changes in the distance of less than infrared spectroscopy, no difference value the sensor insensitive or does not work;
- The dual direction of sensor should be installed parallel as far as possible in inline with human movement. In order to increase the sensor angle range, the module using a circular lens also makes the probe surrounded induction, but the left and right sides still up and down in both directions sensing range, sensitivity, still need to try to install the above requirements.

HC-SR501 PIR MOTION DETECTOR



-
- 1 working voltage range :DC 4.5-20V
 - 2 Quiescent Current :50uA
 - 3 high output level 3.3 V / Low 0V
 4. Trigger L trigger can not be repeated / H repeated trigger
 5. circuit board dimensions :32 * 24 mm
 6. maximum 110 ° angle sensor
 7. 7 m maximum sensing distance

Product Type	HC--SR501 Body Sensor Module
Operating Voltage Range	5-20VDC
Quiescent Current	<50uA
Level output	High 3.3 V /Low 0V
Trigger	L can not be repeated trigger/H can be repeated trigger(Default repeated trigger)
Delay time	5-300S(adjustable) Range (approximately .3Sec -5Min)
Block time	2.5S(default)Can be made a range(0.xx to tens of seconds
Board Dimensions	32mm*24mm
Angle Sensor	<110 ° cone angle
Operation Temp.	-15-+70 degrees
Lens size sensor	Diameter:23mm(Default)

Application scope

- Security products
- Body induction toys
- Body induction lamps
- Industrial automation control etc

Pyroelectric infrared switch is a passive infrared switch which consists of BISS0001 ,pyroelectric infrared sensors and a few external components. It can at open all kinds of equipments, including incandescent lamp, fluorescent lamp, intercom, automatic, electric fan, dryer and automatic washing machine, etc. It is widely used in enterprises, hotels, stores, and corridor and other sensitive area for automatical lamplight, lighting and alarm system.

Instructions

Induction module needs a minute or so to initialize. During initializing time, it will output 0-3 times. One minute later it comes into standby.

Keep the surface of the lens from close lighting source and wind, which will introduce interference.

Induction module has double -probe whose window is rectangle. The two sub-probe (A and B) is located at the two ends of rectangle. When human body r to right, or from right to left, Time for IR to reach to reach the two sub-probes differs.The lager the time difference is, the more sensitive this module is. Wh body moves face-to probe, or up to down, or down to up, there is no time difference. So it does not work. So instal the module in the direction in which mos activities behaves, to guarantee the induction of human by dual sub-probes. In order to increase the induction range, this module uses round lens which ca from all direction. However, induction from right or left is more sensitivity than from up or down.