uRAD

Tutorial

Arduino version

Contents

Introduction 1

uRAD Tutorial

This tutorial has been thought for beginners. In it, you will learn the basic information about how to start with your Arduino board and how to use it together with uRAD. If you need further information about Arduino, we recommend reading the book written by Michael Margolis, Arduino Cookbook or visiting the guide in the official website www.arduino.cc/en/Guide/HomePage.

In addition, you will see how to use the Graphical User Interface to learn how to control and configure uRAD in an easier and more intuitive way.

uRAD is an Arduino shield. Therefore, it only works together with an Arduino compatible board. The boards you can use are those that have the usual pins disposition: Arduino Uno, Zero, Leonardo, 101, Mega, Due, M0, Yun and Ethernet, among others.

Additional Information

Released versions:	uRAD v1.0 15/07/2018
Purchase:	www.uRAD.es/en/arduino
Technical specifications:	www.uRAD.es/en
Software download:	www.uRAD.es/en/mi-cuenta/downloads
Contact:	contact@uRAD.es

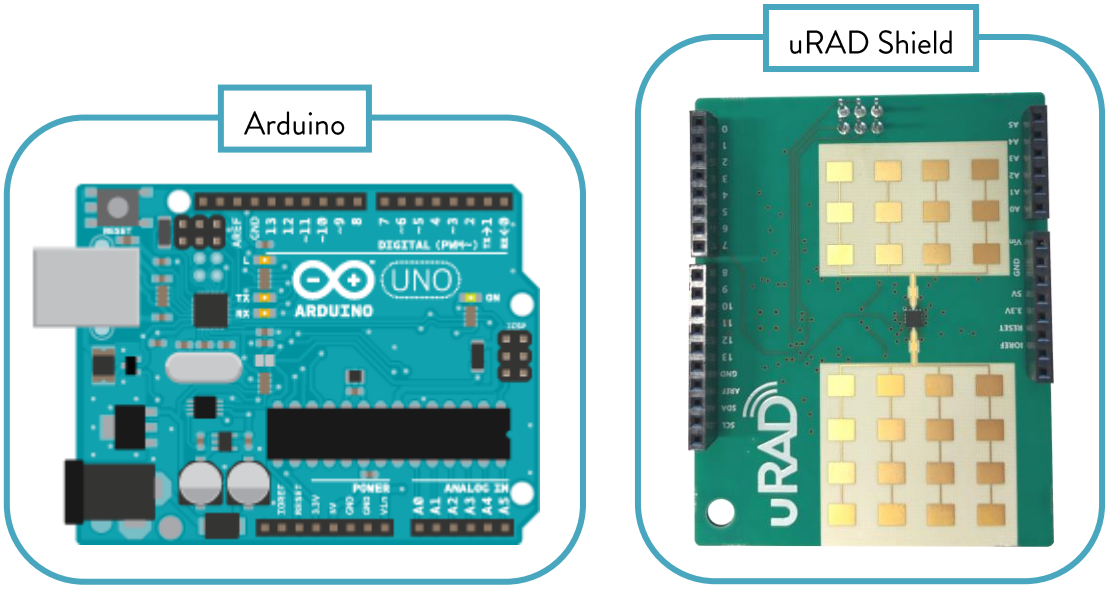
Materials

2

uRAD Materials

The material that you need for the realization of this tutorial is detailed below.

1. You need an **Arduino board**. It can be any of the models with the usual pins disposition, such as Arduino UNO, Zero, Mega, Ethernet, etc.
2. **uRAD Shield**: You can connect uRAD Shield directly to your Arduino using the pins it brings or through cables between the pins used and their corresponding ones in your Arduino.
3. **USB cable**. You will need the corresponding USB cable to connect your Arduino + uRAD to the computer. This cable will have 3 functions: it will be the programming medium of your Arduino, it will transmit the data generated by uRAD to your computer and it will feed Arduino + uRAD.
4. **Charger**: If your application does not require data transfer in real time but it will interact with an actuator when a given event occurs, you can use a charger to power Arduino + uRAD so that your equipment has greater mobility.



USB

Charger

Extra Components

Thanks to the integration of uRAD with the Raspberry Pi programming platform, and its hardware compatibility, it is possible to connect other types of sensors that interact with the measurements provided by uRAD. In this way, it is possible to create complex projects and solve everyday problems in a simple and fast way.

WARNING: It is important that you keep in mind that those extra elements that you use should not cover the vision of the uRAD antennas, because they will influence in your measurement.

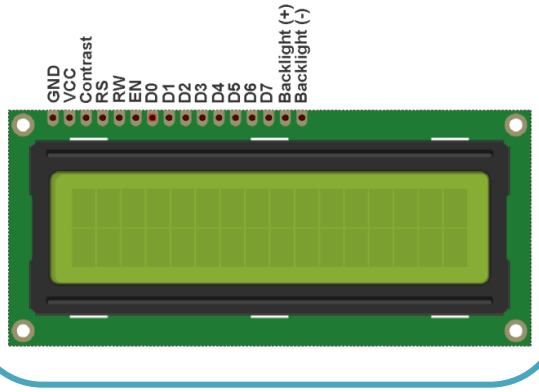
ADVICE: We advise you that if you want to use another type of shield with your Arduino, use uRAD as a module and connect it with Arduino through cables. Read the *User Manual* to know how you should connect uRAD in that case.

Some examples of extra components are:

Leds

Buzzer

LCD screen



Arduino IDE

3

Download Arduino IDE

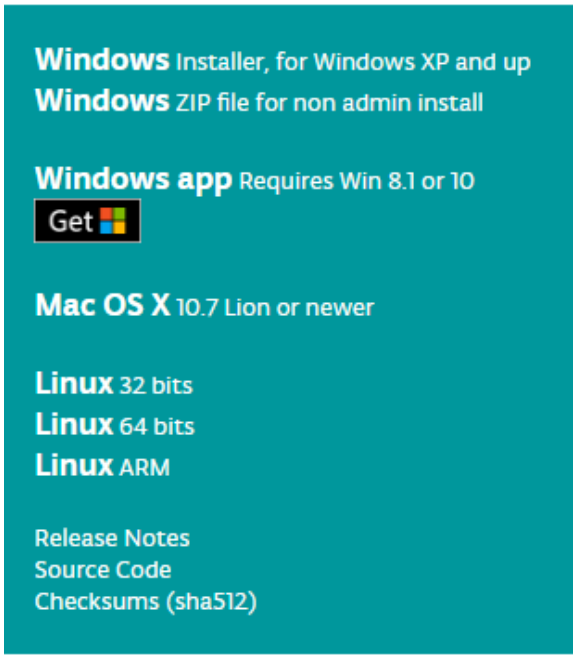
The Arduino Integrated Development Environment (IDE) is the software of the Arduino platform. In this lesson you will learn how to configure your computer to use uRAD together with Arduino in a useful and simple way setting the bases of operation for later lessons.

The Arduino software is available for Windows, MAC and Linux. With it, you will be able to program your Arduino and create all kinds of applications and projects.

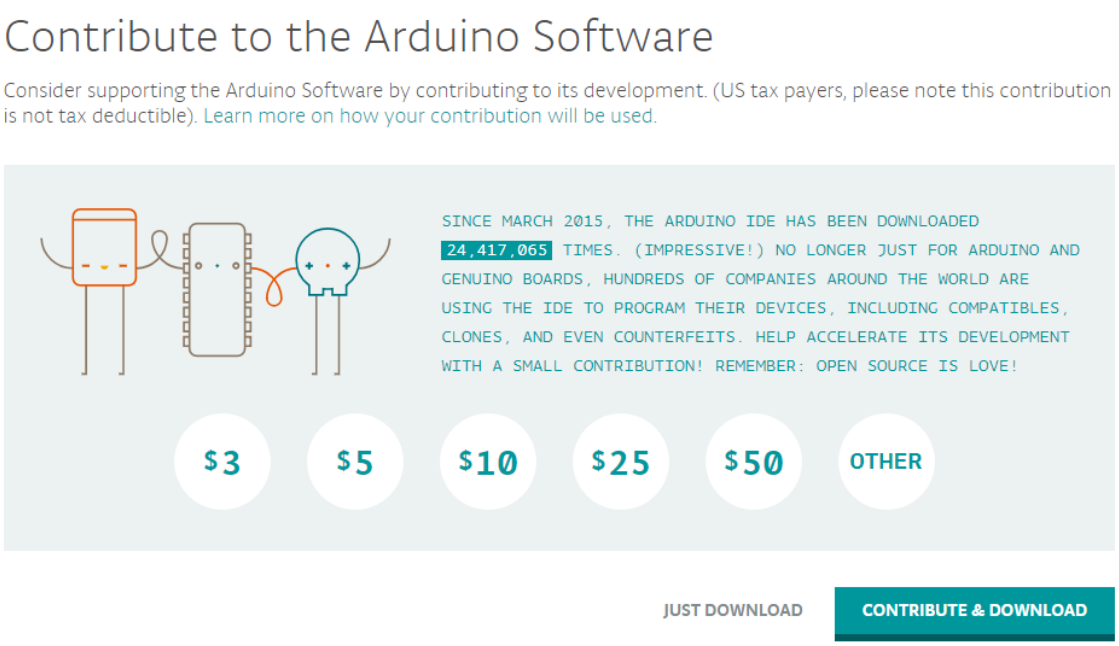
To download the Arduino IDE programming software you must follow the following steps.

Go to the website: http://www.arduino.cc/en/Main/Software.

In it, you will find the different download versions of Arduino IDE. The version available on the website is usually the latest version. Select the download link depending on the operating system that your computer uses. In this tutorial, we select Windows as the operating system.



Once the operating system is selected, in this case *Windows Installer, for Windows XP and up*, it will show up the next download page.

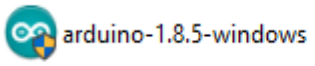


Click on JUST DOWNLOAD to continue downloading the Arduino software without having to contribute economically.

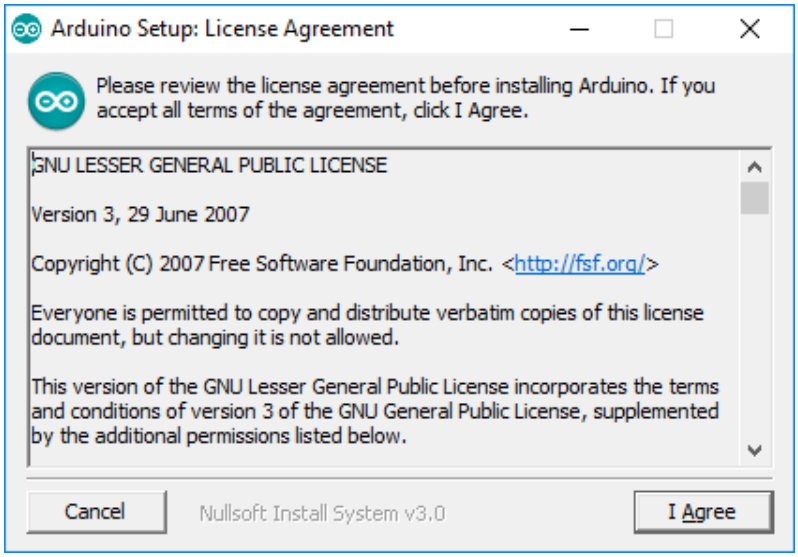
The version available when this practice manual was created was version 1.8.5 and it is the one that we will explain the installation process. This process should not vary significantly from one version to another, so you should be able to follow the explanation without problems.

Install Arduino IDE

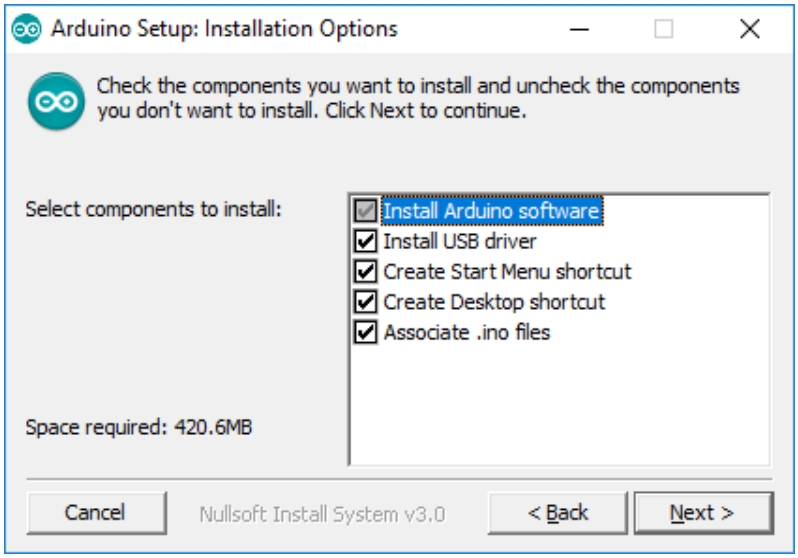
Once downloaded, we install Arduino IDE from the executable.



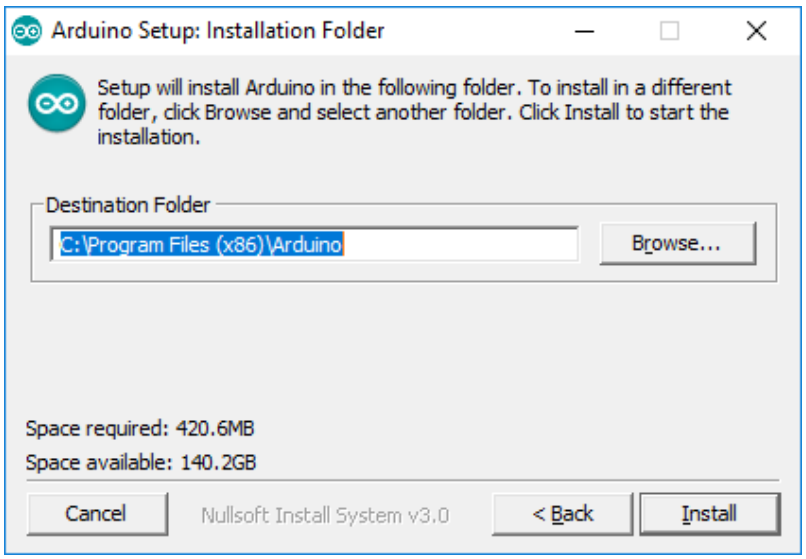
The first thing we must do to install the program is to accept the license terms. To do this, click on *I Agree* in the next window.



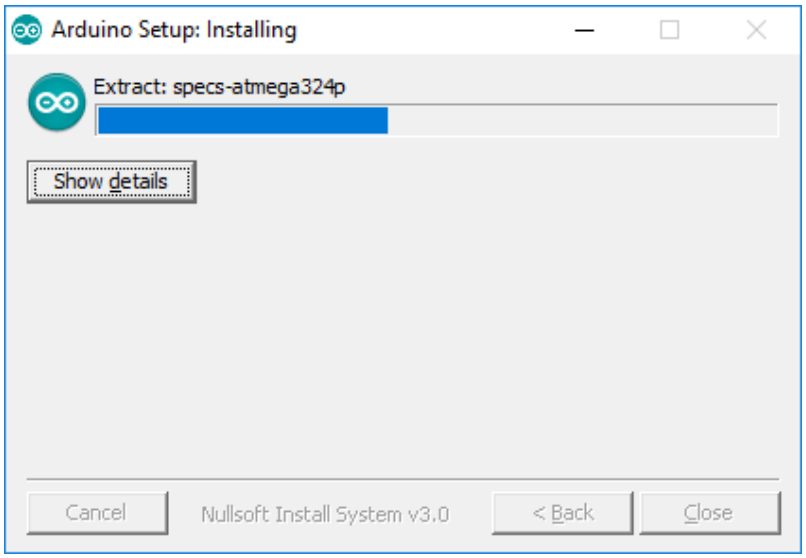
Next, we select the components to be installed and click on *Next*.



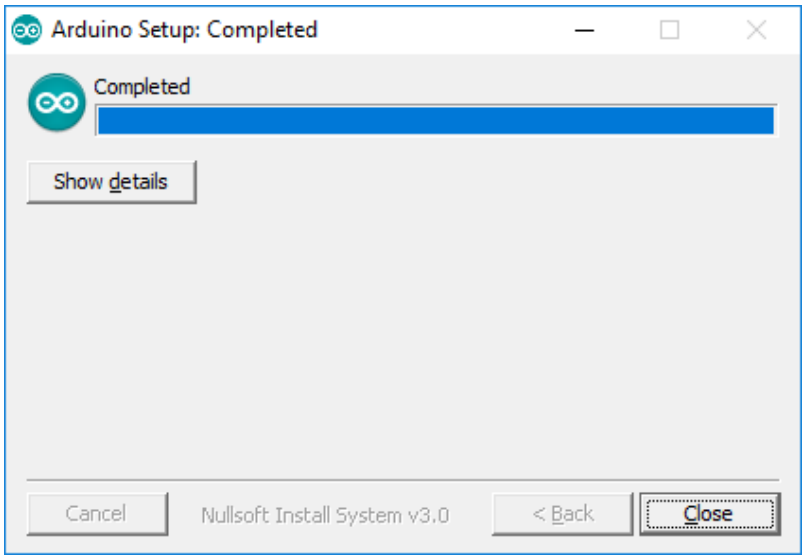
We select the path in our computer where we want to install Arduino IDE, by default it will be installed in *C:\Program Files (x86)\Arduino*



Select *Install* to start with the program installation.



Once finished, close the installation window by clicking on *Close*.

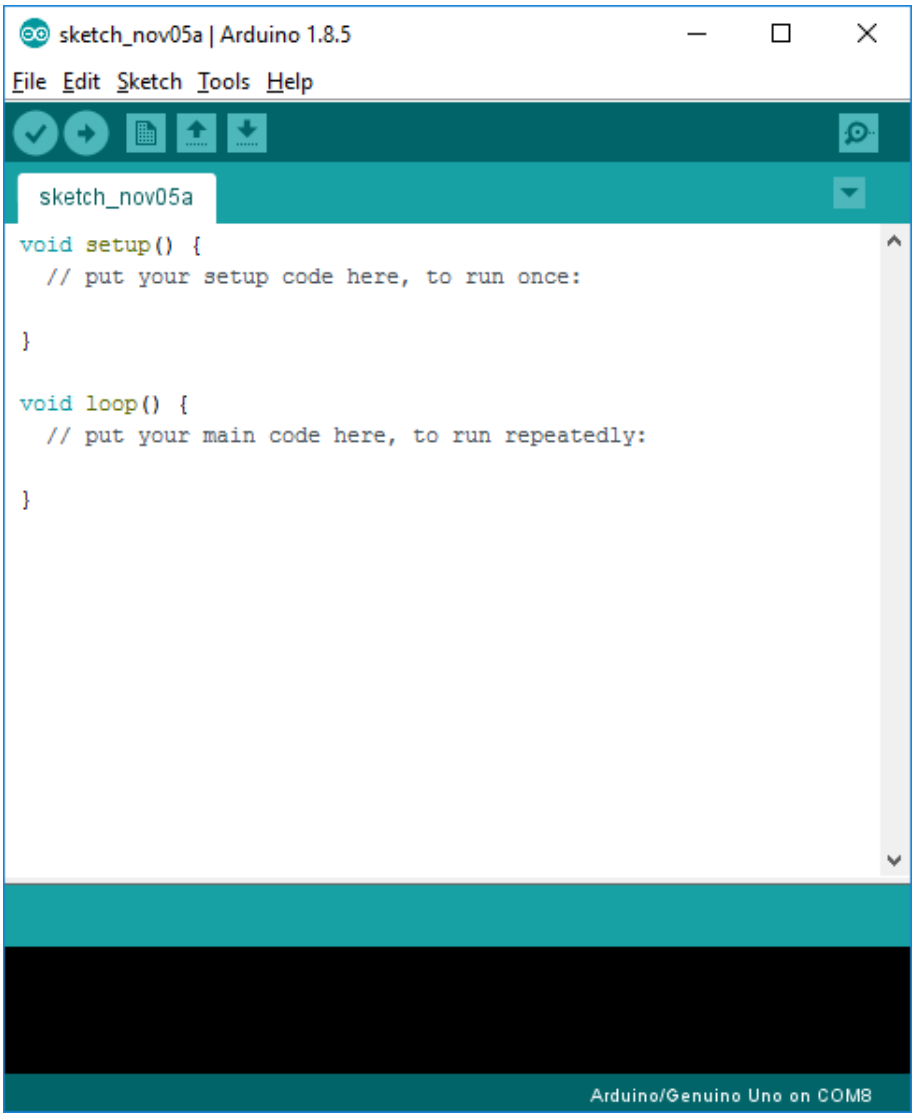


Once the entire process is complete, the Arduino IDE shortcut icon will appear on your desktop.

This completes the process of installing the Arduino IDE development program.

Start Arduino IDE

You have already downloaded and installed Arduino IDE on your computer so the next step will be to start working with it. To do this, double click on this icon to initialize the program and access the Arduino development environment.



Libraries

4

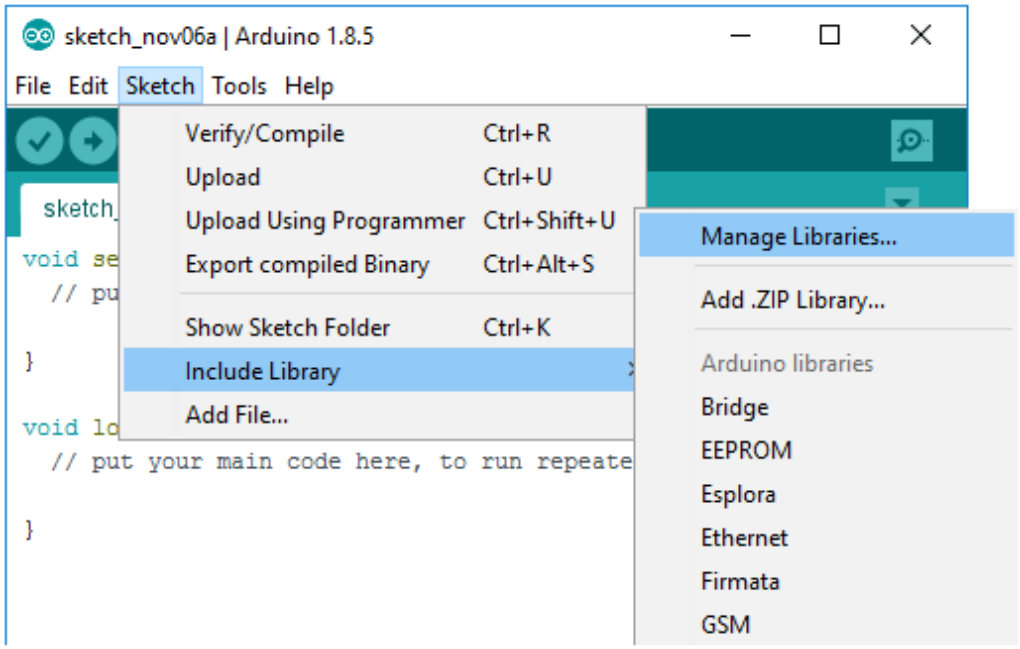
Libraries Installation

Once you have installed the Arduino software, along with the basic functions already integrated in this software, you will have to install the necessary libraries for the communication and programming of uRAD. In addition, many of the extra sensors and add-ons you want to use along with uRAD will need to use extra libraries. This lesson will guide you to learn how to install any type of library from the Arduino database or externally using a compressed file.

Libraries are a collection of code that makes it easier to connect a sensor, screen, module, etc. There are hundreds of additional libraries available on the internet for download that must be installed in order to be used.

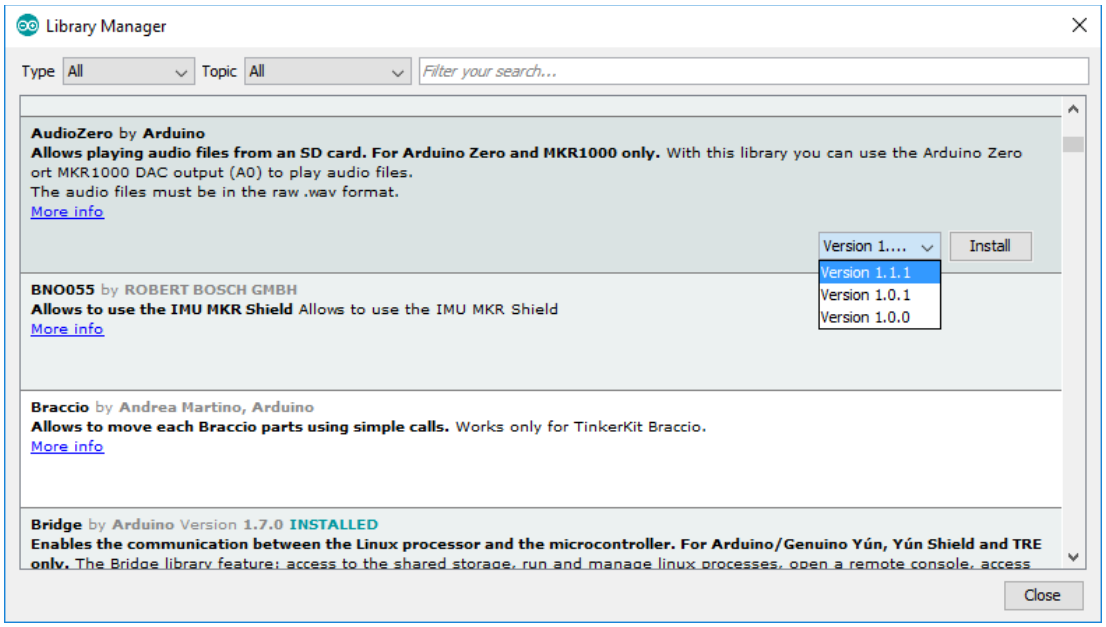
It is possible to install a new library using the library manager of Arduino IDE. Next, we will show an example.

Open the Arduino IDE program and select the tab *Sketch*. Once there, select *Include Library* > *Manage Library*.

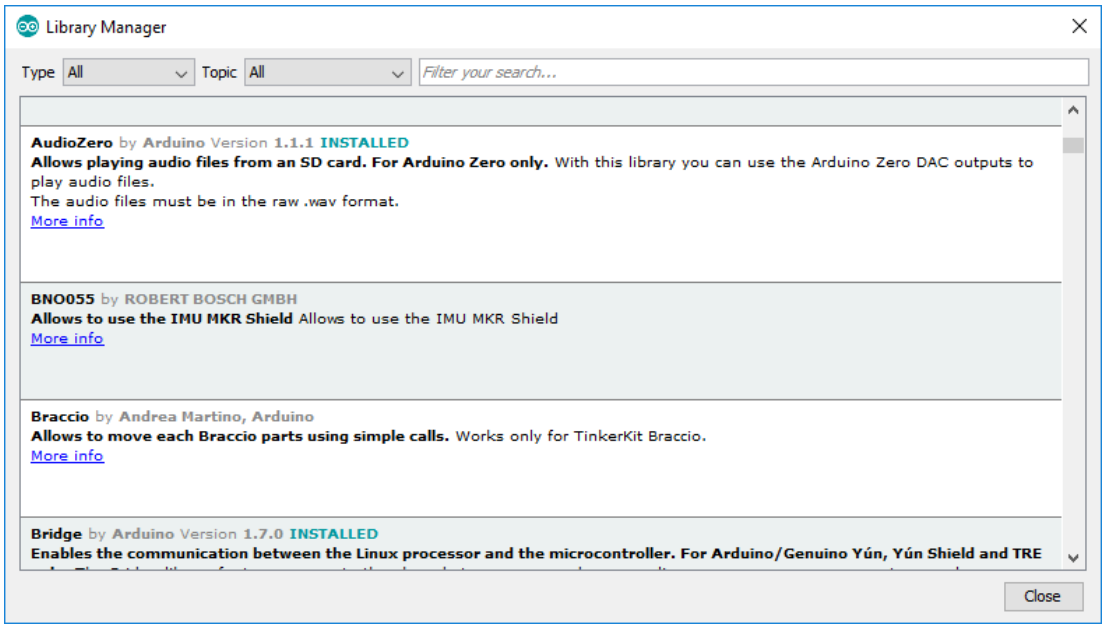


Once this is done, the directory of the libraries will open and you will get a list of libraries already installed or ready to do so. As an example, we are going to install the *AudioZero* library made by Arduino. In many occasions, different versions will be available to install from the same library and in other occasions, only a single version will be available. This can be seen in the version selection menu.

We select the version that we want to install as shown in the following image.



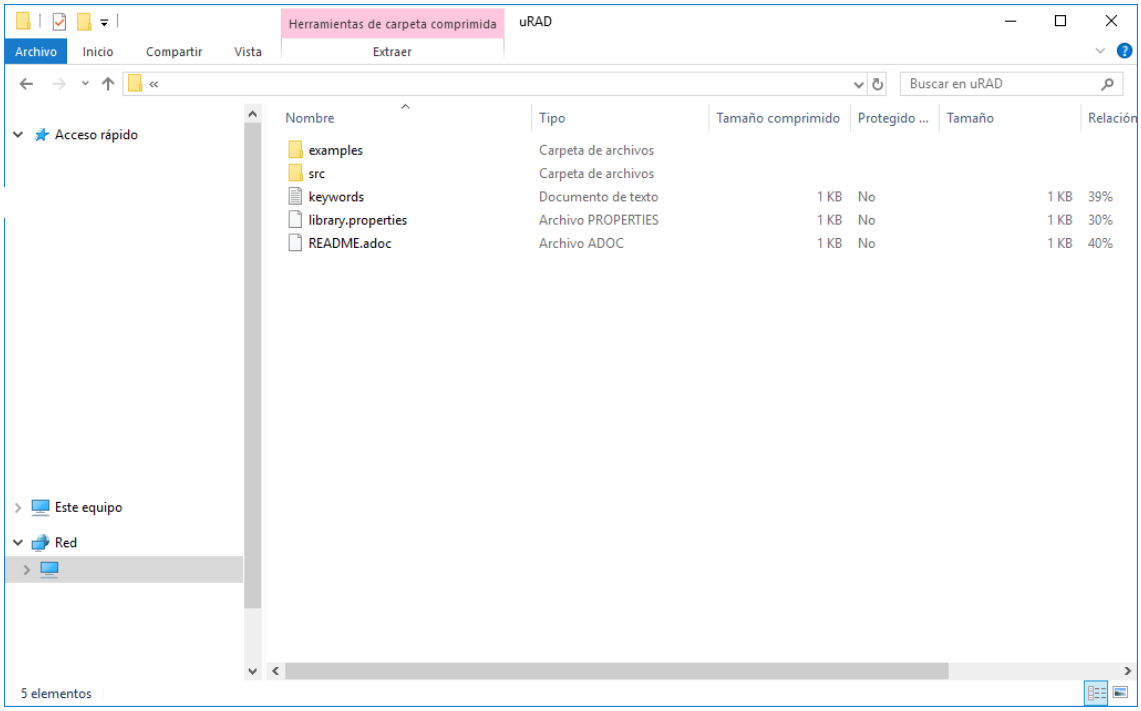
Once this is done, we click on *Install* and the process will begin. This process is usually very fast and does not require restarting the program.



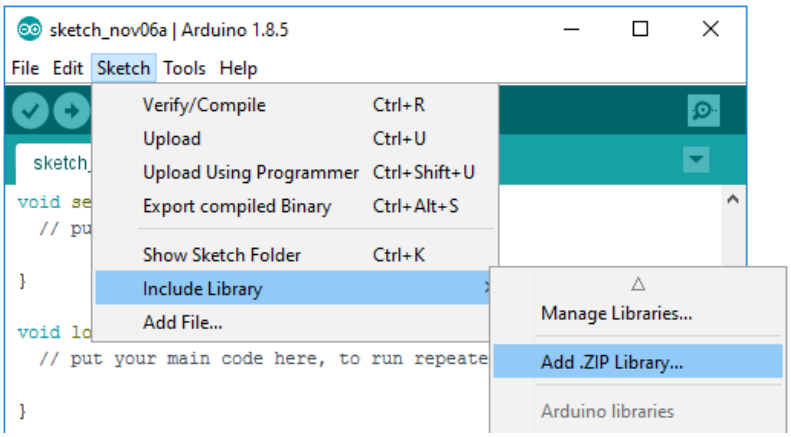
Once done, *INSTALLED* will appear next to the installed library.

uRAD Library Installation

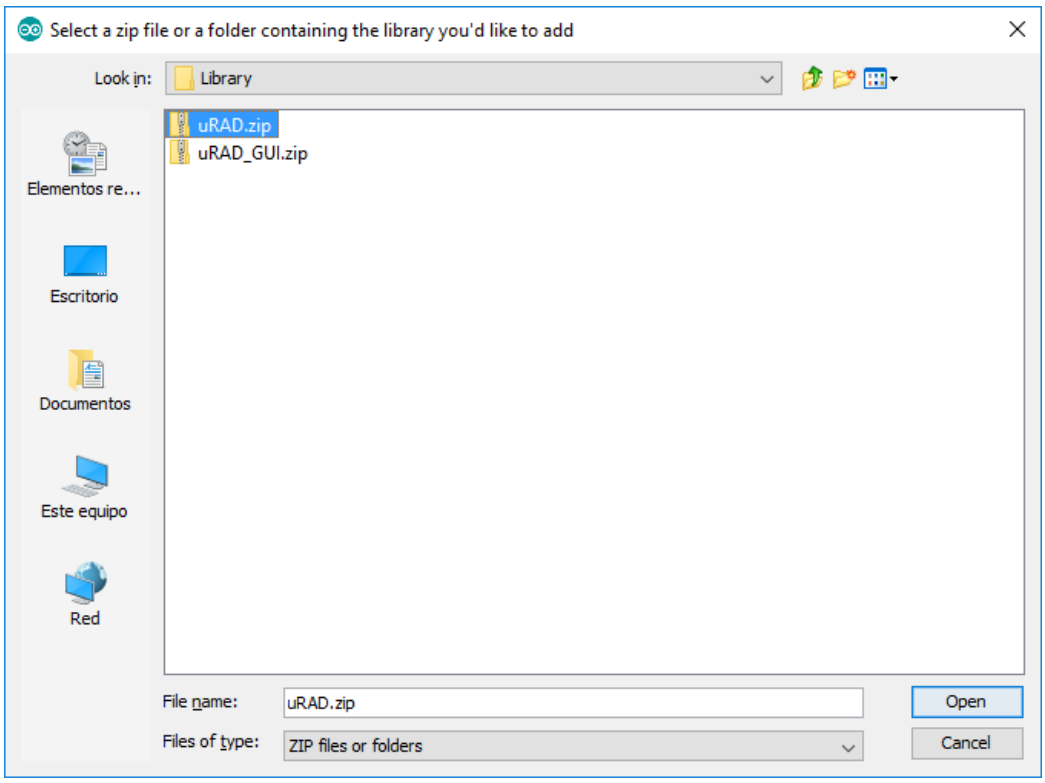
Once we have learned to manage the accessible libraries from Arduino IDE, we will learn to install the necessary library for uRAD. In this case, we will use a compressed ZIP file to import the library. Download from www.urad.es/en/mi-cuenta/downloads/ the folder ***Arduino_software.zip*** corresponding to your operating system and save it in your hard disk. Unzip it and inside *Arduino_software\Library*, you will find the library ***uRAD.zip***. Inside the folde there will be a folder named *src* whit a .cpp and .h files with de source code of the library, an *examples* folder with examples of use and several files (*keywords*, *library.properties*, *README*) with library properties.



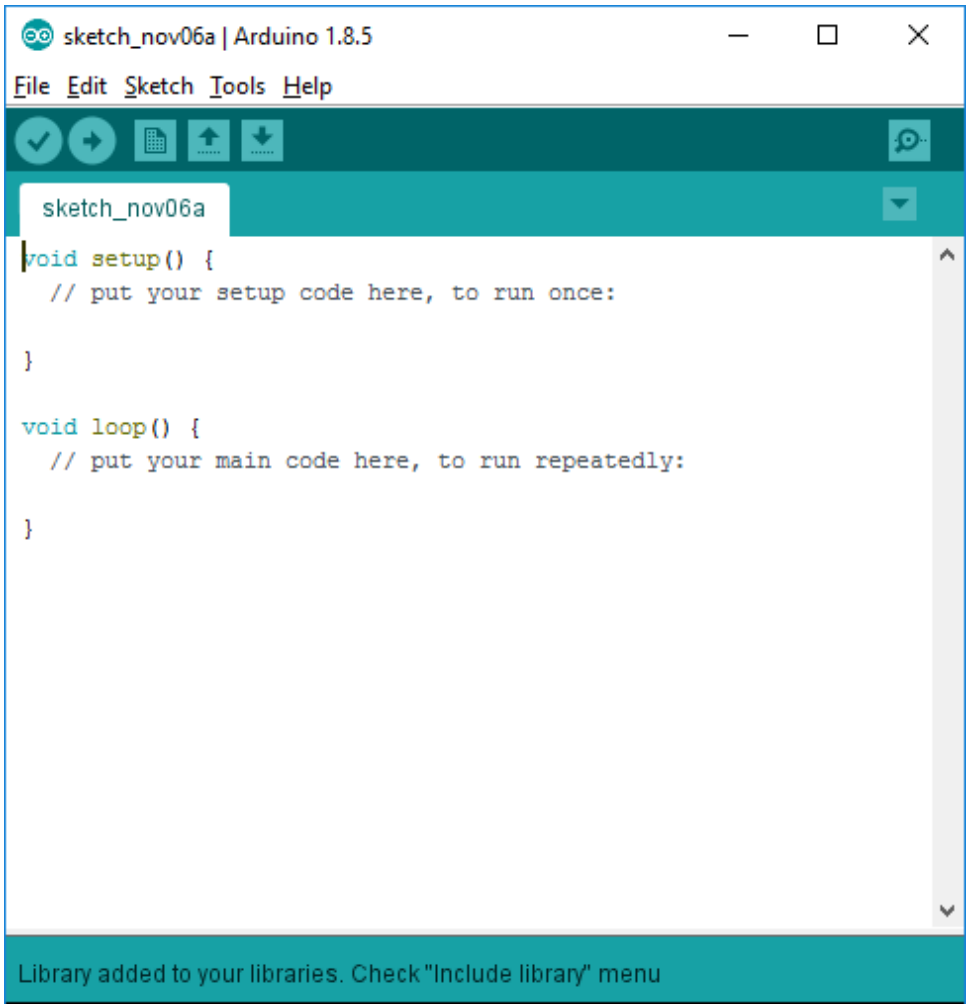
To add the library from uRAD to IDE we will go to *Sketch* > *Include Library* > *Add .ZIP Library...*



You will be asked to select the library that you want to add. To do this, go to the location where you have the file ***uRAD.zip*** and then select *Open*.

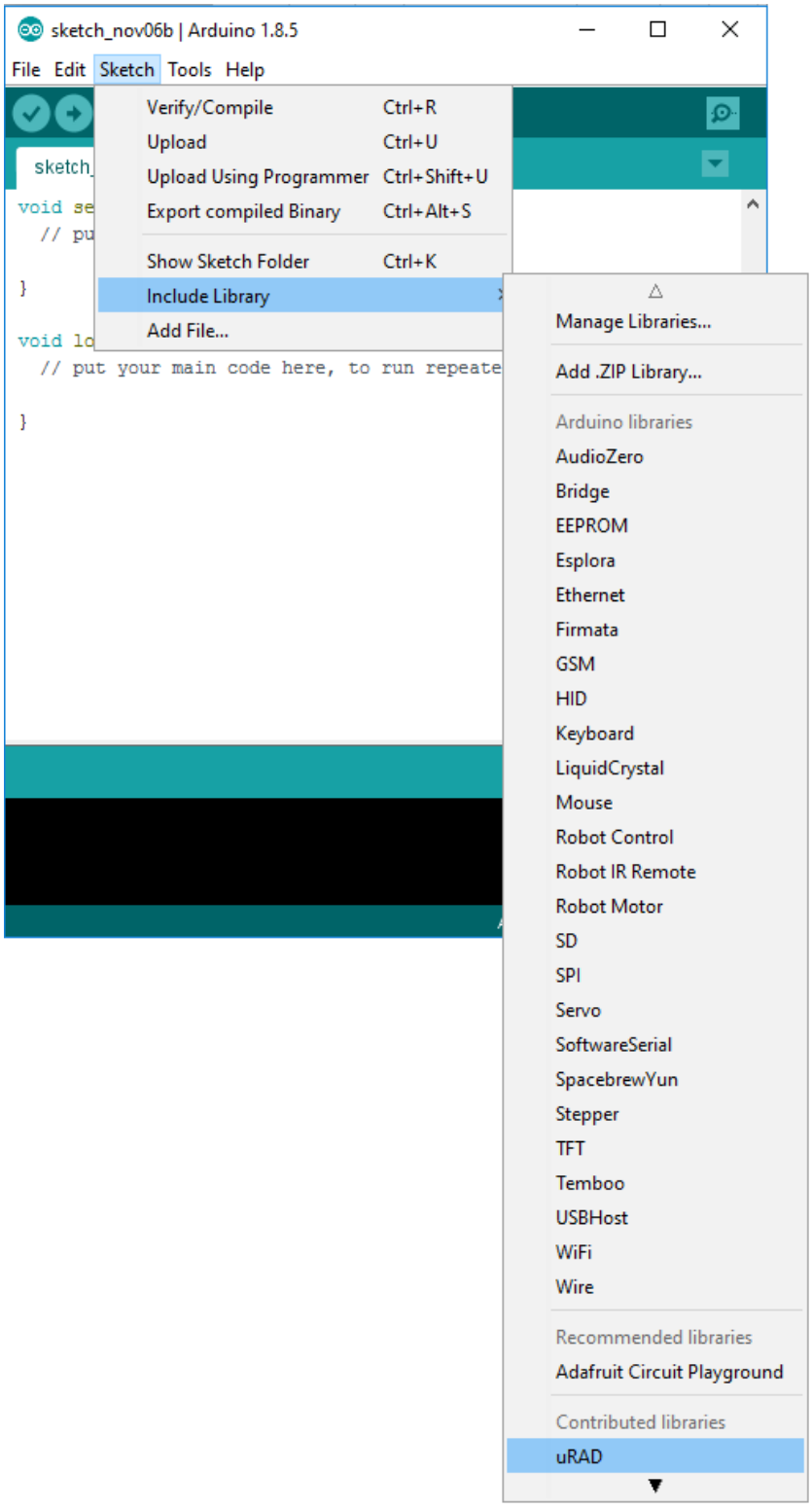


Once this is done, Arduino IDE will notify you its correct import.



You can identify it within the available libraries in *Sketch* > *Include Library*, as shown in the following image.

> *ADVICE:* Keep in mind that to import a library you do not need to restart the program. However, if you want to see the examples of this one, you should do it.



Serial Monitor

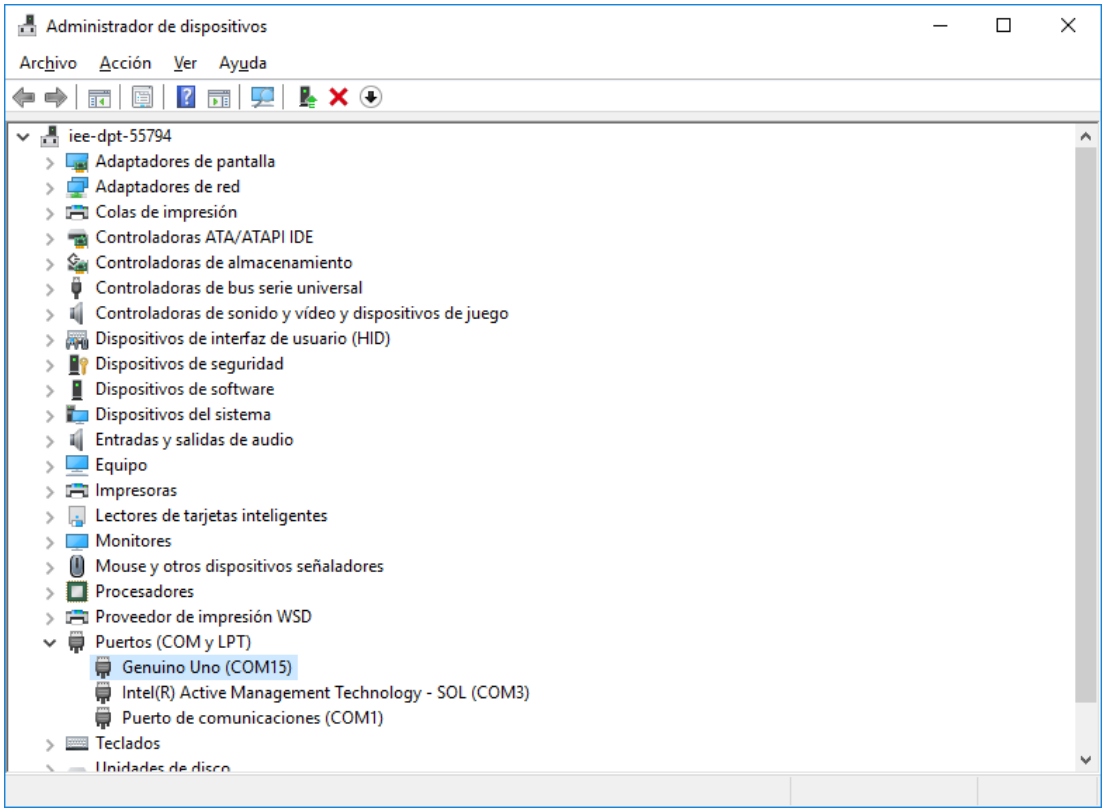
5

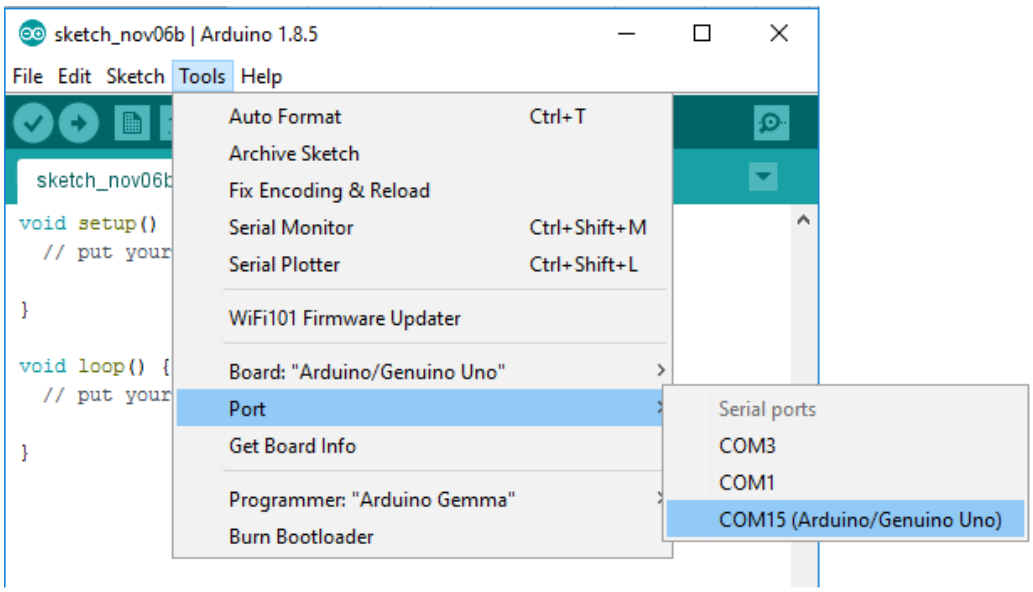
Connection to the Serial Monitor

As we have seen throughout the first lessons, the software of the Arduino platform is IDE (Integrated Development Environment). This software also has a serial terminal associated with it that allows visualizing the data that are sent through the serial port. This terminal is called Serial Monitor.

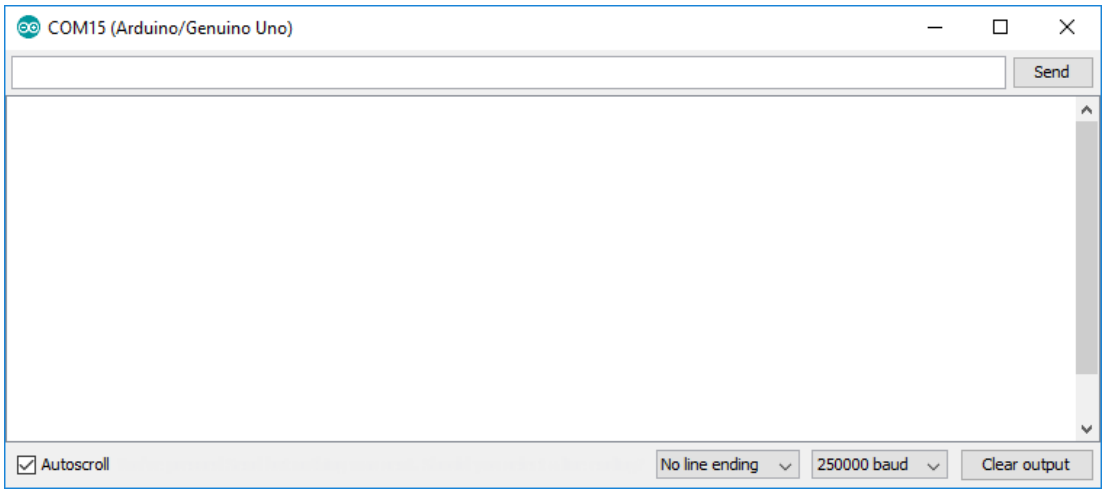
The Serial Monitor is present in any version of Arduino IDE and can be opened by clicking on the icon that appears in the upper right part of the interface.

Once opened, it is necessary to select the port where Arduino will load the code. To do this, we will go to *Tools* > *Port:* and we will select the port where we have Arduino connected. It is important at this point, to select the same COM port where Arduino appears in the Windows Device Manager.

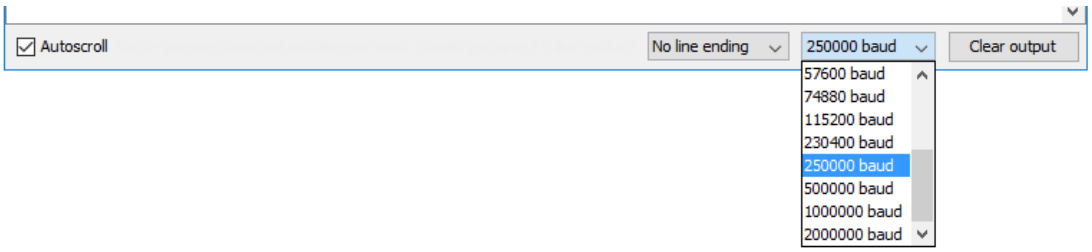




In our case, the COM15 port is where our Ardunio UNO board is connected. Once connected, we proceed to open the Serial Monitor in Arduino where we will have the next window.



This window has different variables to configure, such as the automatic scrolling of the data displayed on the screen (Autoscroll) or the transfer speed. This last value will be set at 250000 baud in order to achieve an efficient data transmission between Arduino and uRAD.



uRAD

6

Basic Aspects

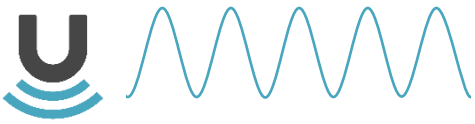
In this lesson, we are going to see a little review of how uRAD works. To learn and deepen the basic operation principles of uRAD, you must refer to the *User Manual* or to www.urad.es. In it, you will find all the information you need to know to understand what a radar is, how it works and what parameters can be measured thanks to it.

As a reminder, we will say that a radar is a device that detects elements that are within its action radius. This equipment works by emitting an electromagnetic wave that is transmitted through the air until it reaches the object/s in question where it is reflected, starting its way back to the radar. Depending on the shape of the transmitted wave as well as the radar architecture, the type of radar is determined.

uRAD can detect up to 5 different elements that are in its field of view. In addition, uRAD allows you to know the distance to the target, as well as its relative radial velocity. Finally, uRAD also provides the amount of reflected power, which can be used as an estimation of the object size or detection level. It is important to know that, if sufficient power is not reflected, the element is not detected. Therefore, very small or distant targets will not be possible to detect them.

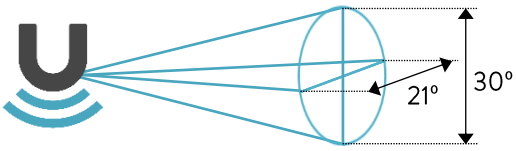
Remain that the main features of uRAD are:

Frequency Bandwidth

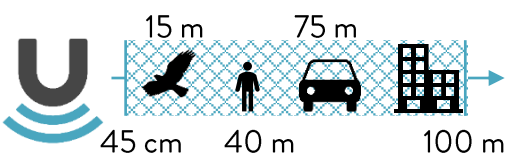


24.005 – 24.245 GHz

Field of View



Distance Range



Accuracy = max. ±4 cm or 0.3%

Velocity Range

Accuracy = ±0.05 m/s

It is important to keep in mind that the distance/velocity range and the accuracy depend on the uRAD configuration and, therefore, will vary depending on the selected operating mode.

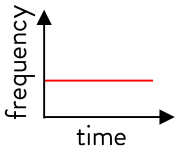
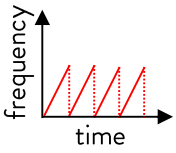
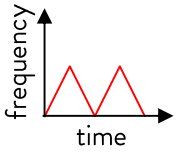
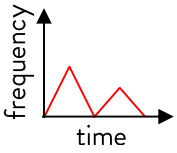
Configuration Parameters

You must enter 8 parameters to configure uRAD. Next, we will explain what parameters are these in order you to learn which configuration is more convenient for your application.

> *REMINDER:* You have much more information readily available in the uRAD *User Manual*.

1. Mode

There are four operation modes that corresponds to 4 different transmitted waveforms: continuous wave (CW), sawtooth, triangular and dual rate. In CW mode, a single frequency is transmitted. This is the most common mode used by Doppler radars. In the rest of the modes, a frequency ramp is transmitted, also called frequency modulated continuous wave (FMCW). Each mode has their advantages and disadvantages and depending on the application, you will select one or another. Next table summarizes the main features of each mode.

Mode	1	2	3	4
Name	CW	Sawtooth	Triangular	Dual Rate
Waveform				
Measured parameters	Velocity	Distance	Distance Velocity	Distance[1] Velocity[1]
Movement detector	YES	YES	YES	YES
Distance range (m)	0 to 60	0.45 to 100	0.45 to 100	0.45 to 75
Distance accuracy (m)	-	Max: ±0.3% ±0.04	Max: ±0.3% ±0.04	Max: ±0.3% ±0.04

Distance resolution[2] (m)	-	1.5	Different[3] velocity or 1.5	Different[3] velocity or 1.5
Velocity range (m/s)	0 to ± 75	-	0 to ±75	0 to ±75
Velocity accuracy (m/s)	±0.05	-	±0.25	±0.25
Velocity resolution[2] (m/s)	3	-	Different[3] distance or 3	Different[3] distance or 3
Update rate (samples/second)	38	21	13	7

- [1] The fact that the dual rate mode uses two different triangular ramps consecutively, provide enhanced results due to ghost targets reduction. **This mode is especially useful in multi-target scenarios.**
- [2] Distance and velocity resolution indicates de minimum distance or velocity that two targets must be separated to be discerned as a single target each one.
- [3] In mode 3 and 4, uRAD can discern two targets at exactly the same distance but different velocity, and vice versa, respectively.
- [4] Update rate is maximum when the number of samples is minimum.

2. f0

It is the operation frequency in CW mode or the ramp start frequency in the others. Since uRAD is configured to operate between 24.005 and 24.245 GHz, you can select f0 from 0 to 250 in CW mode or from 0 to 200 in the others (minimum frequency sweep allowed in ramp modes is 50 MHz).

f0	Mode 1	Mode 2, 3, 4
Range value	5 to 245	5 to 195

3. BW

In ramp modes (modes 2, 3, 4), you must define an operation frequency bandwidth (BW). In other words, the frequency sweep which is varied in every ramp. Depending on the f0 introduced, there will be a larger or lower BW available to select. Minimum value is 50 and in the case you try to introduce a value higher than the available BW, uRAD will select the maximum BW allowed:

$$BW_{max} = 245 - f0$$

BW is a very relevant parameter because defines the accuracy of the system. The higher the BW, the better the accuracy. Moreover, selecting higher BW makes uRAD more capable to distinguish between targets that are very close to each other.

BW	Mode 1	Mode 2, 3, 4
Range value	-	50 to (245 – f0)

Because mode 1 is mono-frequency, the BW here has no sense and the value introduced causes no effect on the configuration.

ADVICE: Although minimum BW is 50, we recommend using the maximum BW available, or at least 150, unless your specific application requires lower BW.

ADVICE: If you have more than one uRAD module detecting the same area, select different f0 and BW for each one to reduce mutual interference.

4. Ns

It defines the number of samples that uRAD takes of the reflected wave to calculate distance, velocity, etc. This parameter is even more important in modes 2, 3, 4 because it also defines the ramps duration and therefore the update rate.

Ns	Mode 1	Mode 2	Mode3	Mode 4
Range value	50 to 200	50 to 200	50 to 200	50 to 200
Update rate [samples/second]	38 to 21	21 to 15	13 to 9	7 to 5

ADVICE: In mode 1, selecting higher Ns makes uRAD able to better discern between targets that have very similar velocity.

It seems that it is always better to select the lowest Ns to have the best update rate. However, the relation between BW and Ns also determines the theoretic maximum distance that uRAD can see.

$$Distance_{max} = 75 \times \frac{Ns}{BW}$$

Therefore, the lower maximum distance (Ns = 50, BW = 250) is only 15.625 meters whereas the higher maximum distance (Ns = 200, BW = 50) is 300 meters.

WARNING: although the theoretical maximum distance can be up to 300 meters, uRAD does not emit enough power to see targets at 300 m, because it must accomplish the legal limits. So that, the actual maximum detection distance is around 100 meters.

5. Ntar

It is the maximum number of targets to detect. A maximum of 5 targets can be selected. If more than 5 elements are in the field of view, uRAD gives you the information of the 5 most significant ones that reflect more power.

Ntar	Mode 1, 2, 3, 4
Range value	1 to 5

6. Rmax / Vmax

Rmax is the maximum distance where the targets will be searched. Rmax is independent of the theoretical maximum distance, which defines the detection range. With Rmax you establish the length of the zone you want to detect, that is, the action radius of uRAD.

For example, imagine that you fix BW = Ns = 100 that gives you a theoretic maximum distance equal to 75 m. If you define Ntar = 3 and Rmax = 20, uRAD will give you de information of 3 targets that are located between 0 and 20 meters, independently that uRAD detects more relevant targets in the range from 20 to 75 meters.

On the other hand, in mode 1, uRAD is not able to obtain distances. Therefore, Rmax here means Vmax, the maximum velocity range where the targets will be searched.

Rmax / Vmax	Mode 1	Mode 2, 3, 4
Range value	0 to 75	1 to 100

If you select Rmax = 100 in mode 2, 3 or 4, uRAD will search targets in the range defined by the maximum distance formula. So you can detect targets farther than 100 meters.

7. MTI

It activates or desactivates the Moving Target Indication (MTI) mode. **In this working mode, all static objects are ignored, and only the information of targets that are in movement respect to uRAD is provided.** This feature is only usually available in high performance radars due to the complexity in the data processing. Because of that, it is necessary to define a suitable sensitivity for each scenario.

A value equal to 0 indicates that MTI mode is disabled, and therefore, the information of ALL targets, static or not, is given. A value equal to 1 indicates that the mode is activated and therefore, the static targets are ignored and their information is omitted.

MTI	Modo 1	Modo 2, 3, 4
Range value	-	0 (disabled), 1 (activated)

In mode 1, MTI has no relevance because, by default, this mode is only for moving targets.

ADVICE: This mode can be very useful indoor, for instance, where there may be lot of no relevant static targets and it is only desired to measure the moving ones.

8. Mth

It defines the sensitivity of uRAD when it is working as a movement detector. As you will see in the next section, uRAD alerts you when it detects that some target is moving in its detection area, whenever you want this alert.

With Mth, you can define up to 4 detection thresholds. This threshold is defined as function of target reflectivity which is proportional to the size of the target and inversely proportional to its distance (bigger and closer targets reflect more). The reflectivity of a target also depends on its material. For instance, a metal object will reflect more than a glass object. In addition, uRAD is able to cross many other materials such as glass, pane, plastic, etc., so you can detect moving targets through various materials, such as a plasterboard wall.

Mth = 4 makes uRAD extremely sensitive to any reflectivity, whereas Mth =1 means that only very big or close targets active the alert.

Mth	Mode 1, 2, 3, 4
Range value	1 (low) to 4 (high) sensitivity

ADVICE: Try several values of Mth to find the value that better fits your particular scenario. In this way, you will reject undesired alarms.

WARNING: MTI and Mth are independent. You can define MTI = 0 for receiving the information of both static and moving targets, and at the same time, obtaining the movement alert with Mth.

Detected Information

uRAD provides you with complete information of its detection range:

- Distance: returns the distance from uRAD to each detected target.
- Velocity: returns the radial relative velocity between uRAD and each detected target.
- SNR: returns the Signal to Noise Ratio of each detected target. This gives you an idea of the amount of reflected signal of each target, and therefore the size and reflectivity of the target. Technically speaking, it is the difference in magnitude between the reflected signal due to the target and the noise floor due to the whole system. **SNR will hardly exceed a value of 40, in any scenario.**
- Movement: returns TRUE or FALSE if movement of any target IS or IS NOT detected.
- I, Q: returns the total reflected signal decomposed in two arrays with the In-phase and Quadrature components, for advanced data signal processing.

Available returned information depends on the configuration mode.

	Mode	Unit
Distance	2, 3, 4	meters
Velocity	1, 3, 4	meters/seconds
SNR	All	dB
Movement	All	TRUE/FALSE
I, Q	All	Arbitrary units from 0 to 4095

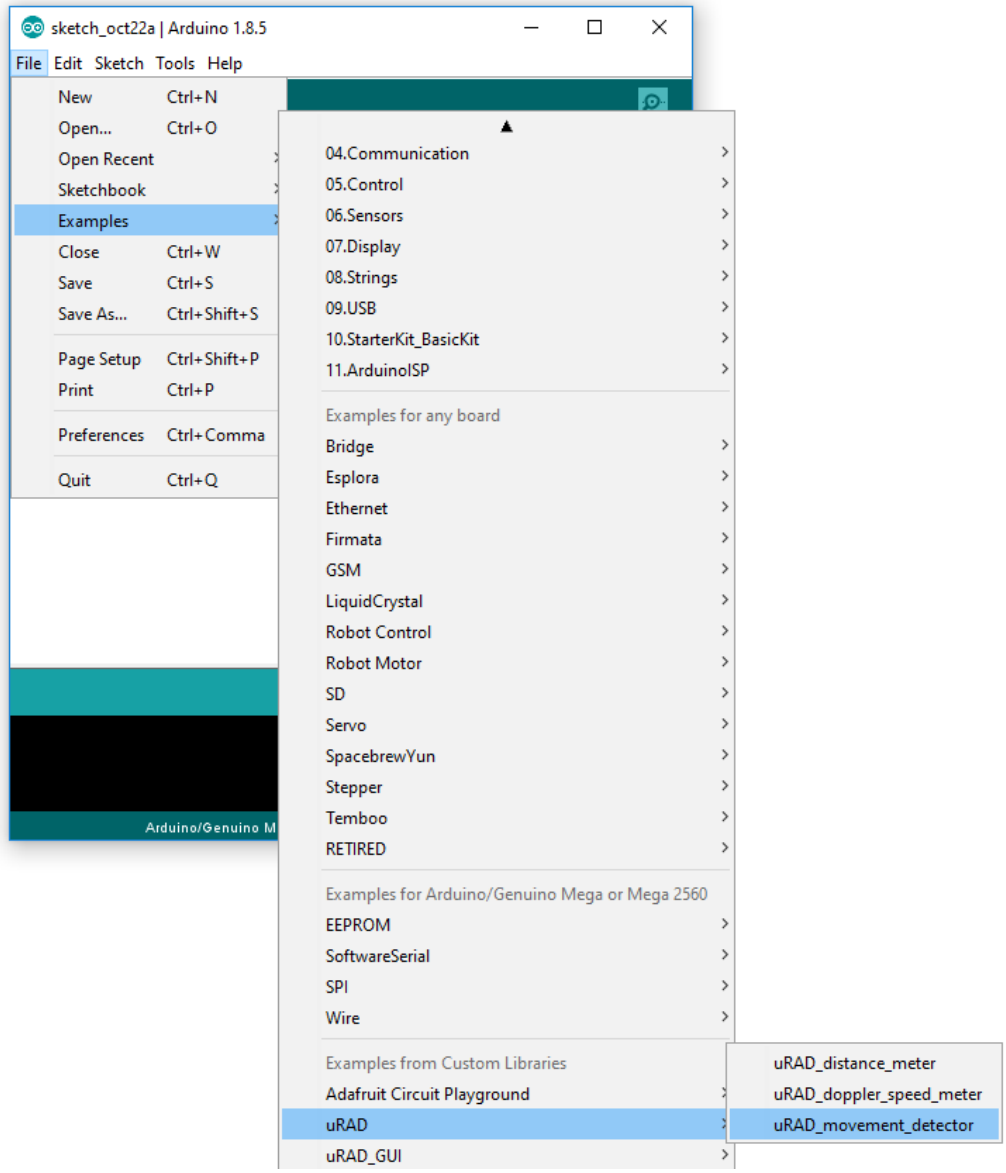
Example 1

7

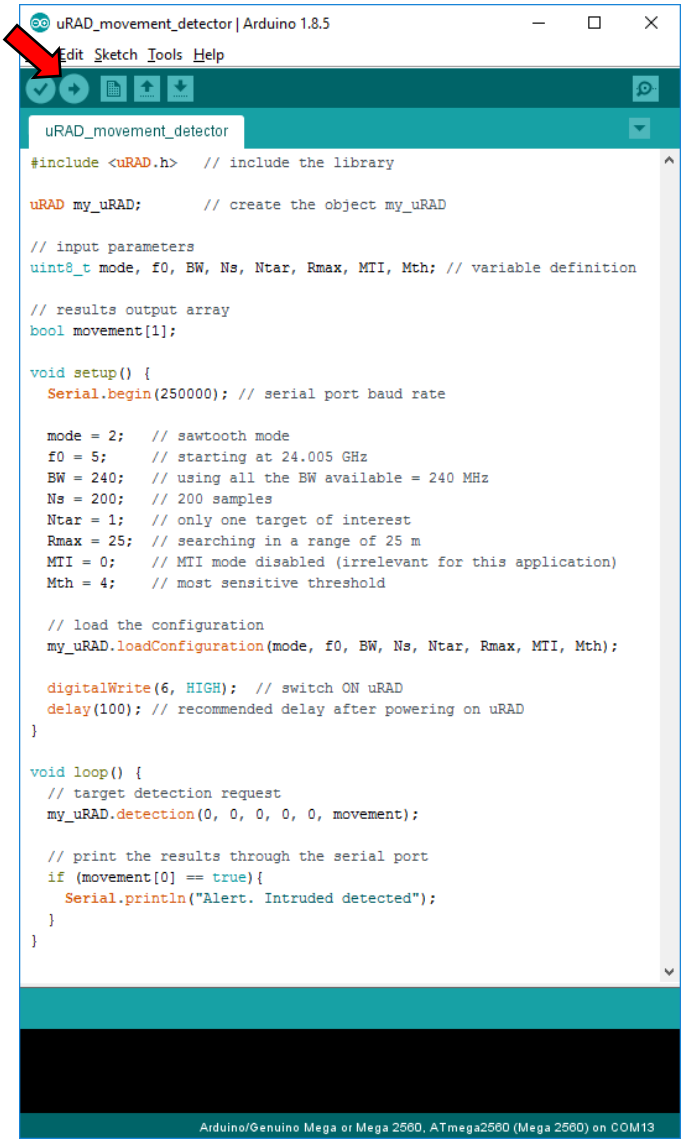
Movement Detector

The first example we are going to do is a movement detector. This example can be found in the examples folder of the uRAD library. To open it you must follow the following steps.

Go to *File > Examples > uRAD > uRAD_movement_detector*



Once this is done, the Arduino program will appear to configure uRAD as a motion detector. To upload the program to the card, just click on the button with an arrow pointing to the right, in the upper left.



One of the simplest applications is to use uRAD as a movement detector. In the following code, we print a warning message in the Python console when a movement is detected.

Now let see step by step the different lines of code used to program uRAD as a movement detector.

WARNING: Along this analysis, it is important to take into account that reserved words appear in red, while green words are generic names that can be chosen by the user.

The first thing we must do in any uRAD program is including the library. To do this, we use the following command.

```
#include <uRAD.h> // include the library
```

This is necessary because Arduino and uRAD must communicate and work together. To perform this communication, this library has been programed.

REMINDER: We have included this library in previous lessons, so it must be in your Arduino IDE. Recuerda que hemos incluido esta librería en lecciones previas por lo que deberá estar en tu Arduino IDE. If not, you must include it so that uRAD works. Go to Chapter 4.

The next step will be to create the uRAD class object. This will allow you to use the two available functions that control your uRAD shield with a specific name. To define this object, the following command is used.

```
uRAD my_uRAD; // create the object with the name my_urad
```

Now, you can use the functions `my_uRAD.loadConfiguration(…)`, and `my_uRAD.detection(…)` that are necessary for running uRAD.

The next thing you need to do is define the variables that will be used in the program. You have input variables (f0, BW, Ns, Ntar, Rmax, MTI, Mth) and output variables (movement).

```
// input parameters
uint8_t mode, f0, BW, Ns, Ntar, Rmax, MTI, Mth;

// results output array
bool movement[1]; // always declare the output array
```

The input variables must be defined as integer numbers of 8 bits (*uint8_t*). The *movement* variable must be definied as an array of length 1 of type *boolean*, becasue the movement detection will be true or false, it tells us if there is or not movement.

The next step will be to define the *baud rate* of the serial port. For uRAD to work optimally we will choose 250,000.

REMINDER: You can go to Chapter 2 to remember how to modify the *baud rate*.

```
void setup() {
  Serial.begin(250000);  // serial port baud rate
```

Within the uRAD configuration, there are 8 parameters that can be modified. All of them must be included in the uRAD configuration function, as we will see below. These parameters have been defined as integers of type *"byte"* (*uint8_t*) in the previous step. Now, what you should do is selecting the value you want them to have depending on the

application you want to have. The following table summarizes the values that can be entered.

mode	1	2	3	4
f0	5 to 245	5 to 195	5 to 195	5 to 195
BW	-	50 to (245 – f0)	50 to (245 – f0)	50 to (245 – f0)
Ns	50 to 200	50 to 200	50 to 200	50 to 200
Ntar	1 to 5	1 to 5	1 to 5	1 to 5
Rmax	0 to 75	1 to 100	1 to 100	1 to 100
MTI	-	0 or 1	0 or 1	0 or 1
Mth	1 to 4	1 to 4	1 to 4	1 to 4

WARNING: Entering a forbidden value in a parameter will result in its default value. Default values are: mode = 3, f0 = 5, BW = max available, Ns = 200, Ntar = 1, Rmax = 75 or 100, MTI = 0, Mth = 4.

- Include this function at least one time, always before your first target detection.
- The configuration does not change until you call again the function.
- Include this function, in any part of you code, as many times as you want to update the configuration for the next target detection.

The values of the variables chosen for this example, for uRAD to function as a movement detector, are the following::

```
mode = 2;     // sawtooth mode
f0 = 5;       // starting at 24.005 GHz
BW = 240;     // using all the BW available = 240 MHz
Ns = 200;     // 200 samples
Ntar = 1;     // only one target of interest
Rmax = 25;    // searching in a range of 25 m
MTI = 0;      // MTI mode disabled (irrelevant for this application)
Mth = 4;      // most sensitive threshold
```

Now, we are going to load the uRAD configuration. This step is very simple. To do this you must include a simple function in your Arduino code that will select the configuration.

```
// load the configuration
my_uRAD.loadConfiguration(mode, f0, BW, Ns, Ntar, Rmax, MTI, Mth);
```

The next thing we are going to do is turn on uRAD. For this, we will use the Arduino function `digitalWrite`. uRAD está conectado en este caso al pin digital 6, por lo que será este pin el que utilizaremos para encender y apagar uRAD. Una vez encendido uRAD, esperaremos unos milisegundos de seguridad.

```
digitalWrite(6, HIGH);    // switch on uRAD
delay(100);   // recommended delay after powering on uRAD
```

REMINDER: If you want to know more about how to turn on or off uRAD you can read the *User Manual*.

With this last function we finish the *void setup* of Arduino and we would start the *void loop*.

REMINDER: Remember that inside the *void setup* is all the configuration information that is execute only one time. On the other hand, *void loop* have all those instructions that are execute periodically.

Obtaining uRAD detection information is as simple as calling a function. Every time you want to get the information, include in your code:

```
my_uRAD.detection(distance, velocity, SNR, I, Q, movement)
```

This function returns 6 vectors with the information detected. The available information returned depends on the configuration mode that we have selected.

	Modo	Tamaño del vector	Tipo de variable
`distance`	2, 3, 4	5	float
`velocity`	1, 3, 4	5	float
`SNR`	All	5	float
`movement`	All	1	boolean
`I`, `Q`	All	Ns (mode 1, 2) 2*Ns (mode 3) 3.5*Ns (mode 4)	uint16_t

- uRAD must always be switched ON before using `my_uRAD.detection(…)`.
- If the information is not available due to the mode selection or because there is not any detected target, `my_uRAD.detection(…)` returns 0 in the corresponding array.
- Target information is listed in each array position from higher to lower SNR.
- If you do NOT need any of the provided info, type 0 in the corresponding input of `detection` function.

Let us start now with specific part of the program where we tell uRAD what we want to measure/detect. We will use the function `my_uRAD.detection` and select the parameters that we want to obtain, in this case, is `movement`. We will do like this:

```
void loop() {
  // target detection request
  my_uRAD.detection(0, 0, 0, 0, 0, movement);
```

The function will return only the alert if a movement is detected, since the rest of the parameters have been a 0.

ADVICE: Do not ask for I and Q unless you are going to use them, because this involves a large amount of transmitted data from uRAD to Arduino and therefore, the update rate decreases as well as the free RAM memory.

Finally, we will show the results on the screen through the serial prot. For this, we will use the `Serial.println` of Arduino. As a condition, we will ask to show us a warning message if, indeed, movement is detected. For this, we will use the following lines of code.

```
    // print the results through the serial port
    if (movement[0] == true){  // if movement is detecte
      Serial.println("Alerta. Intruso detectado."); // show alarm
    }
  }
```

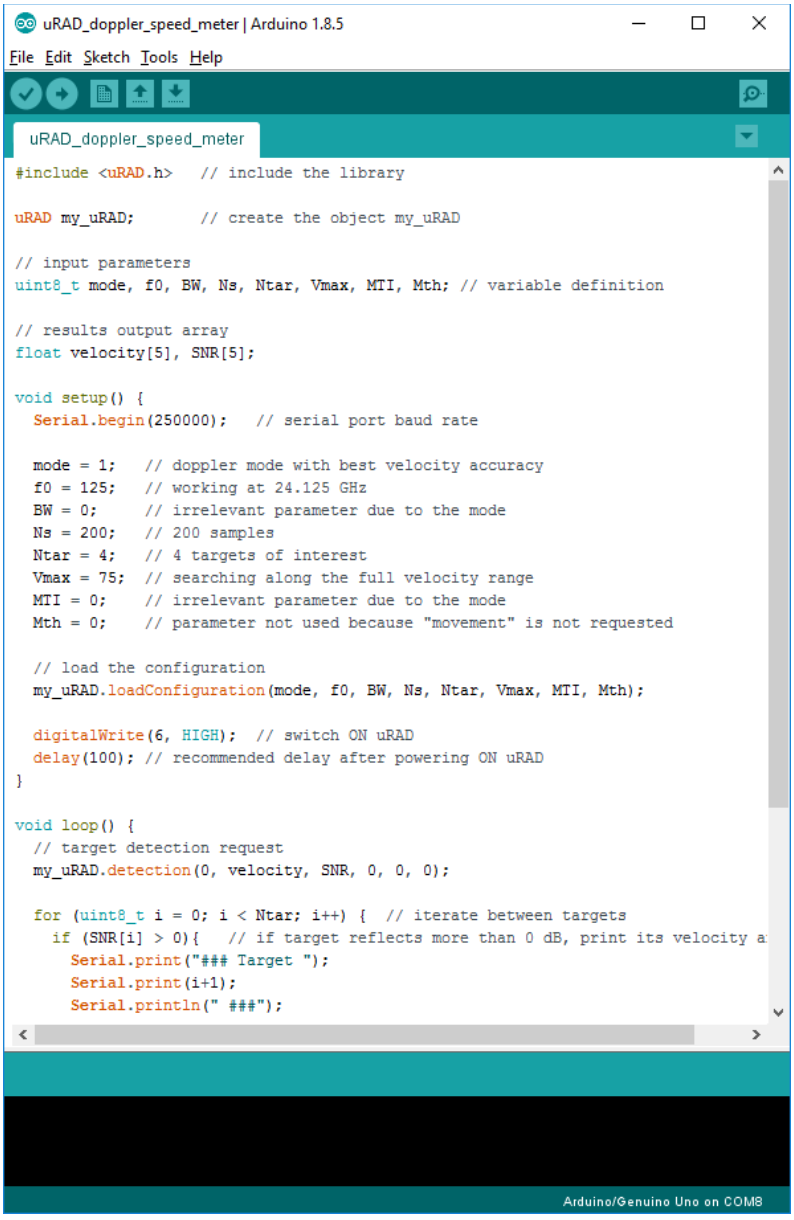
Example 2

8

Doppler Velocity Sensor

In this second example, we are going to program uRAD as a traditional Doppler radar, so it will only be possible to measure the velocity. In addition, we will also request the SNR, in order to know the reflectivity of the detected target and estimate its size accordingly.

To open the example, go to *File > Examples > uRAD > uRAD_doppler_speed_meter*



As a first step, we include the uRAD library and define the object in the same way as was done in Chapter 4.

```
#include <uRAD.h>  // include the library

uRAD my_uRAD;      // create the object my_uRAD
```

Next, we define the input variables needed for uRAD as well as the output variables, those that we want uRAD to measure. In this case, they will be the *velocity* and the *SNR*.

```
// input parameters
uint8_t mode, f0, BW, Ns, Ntar, Vmax, MTI Mth;

// results output array
float velocity[5], SNR[5];
```

ADVICE: Keep in mind that in this case the variable Rmax has been changed to Vmax since the maximum range will be velocity and not distance.

We define the baud rate of the serial port and below the value of each one of the input variables of the uRAD configuration function.

```
void setup() {
  Serial.begin(250000);  // serial port baud rate

  mode = 1;  // doppler mode with best velocity accuracy
  f0 = 125;  // working at 24.125 GHz
  BW = 0;    // irrelevant parameter due to the mode
  Ns = 200;  // 200 samples
  Ntar = 4;  // 4 targets of interest
  Vmax = 75; // searching along the full velocity range
  MTI = 0;   // irrelevant parameter due to the mode
  Mth = 4;   //  parameter  not  used  because  “movement”  is  not
requested
```

Once this is done, we load the desired configuration and turn on uRAD.

```
// load the configuration
my_uRAD.loadConfiguration(mode, f0, BW, Ns, Ntar, Vmax, MTI, Mth);

digitalWrite(6, HIGH);   // switch ON uRAD
delay(100);   // recommended delay after powering ON uRAD
```

Next, we select the parameters we want to measure using the uRAD detection function.

```
void loop() {
  // target detection request
  my_uRAD.detection(0, velocity, SNR, 0, 0, 0);
```

Finally, we print on the screen through the serial port those results that are of interest to us. To do this, we formulate the instruction where we set that the SNR of the detected

target is greater than 0 dB to be shown this objective together with its velocity and SNR value.

```
    for (uint8_t i = 0; i < Ntar; i++) { //iterate between targets
      if (SNR > 0){ // if target reflects more than 0 dB, print its
velocity and SNR
        Serial.print("### Target ");
        Serial.print(i+1);
        Serial.println(" ###");

        Serial.print("Velocity: ");
        Serial.print(velocity[i]);
        Serial.println(" m/s");

        Serial.print("SNR: ");
        Serial.print(SNR[i]);
        Serial.println(" dB");
        Serial.println("");
      }
    }
  }
```

REMINDER: The value of SNR is the ratio between the detected signal level and the noise. In this case, a value of 0 dB has been set, but you can reduce this value in order to detect much weaker movements and play with the sensitivity of uRAD.

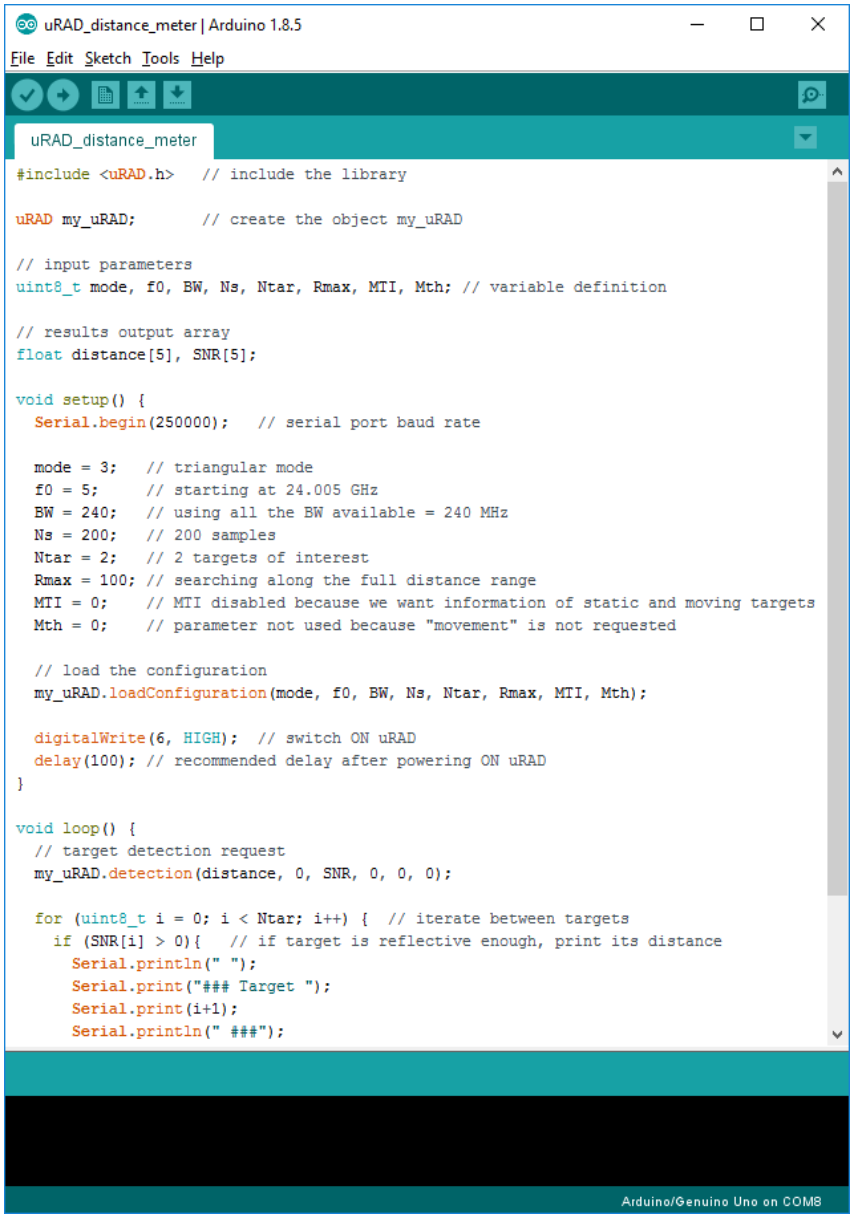
Example 3

9

Distance Meter

In the third example, we will use uRAD as a distance meter. For this, we will use what we have learned in previous chapters.

To open the third example go to *File > Examples > uRAD > uRAD_distance_meter*.



We include the uRAD library and define the object.

```
#include <uRAD.h>  // include the library

uRAD my_uRAD;      // create the object my_uRAD
```

Next, we define the input variables needed for uRAD as well as the output variables, those that we want uRAD to measure. In this case, they will be the *distance* and *SNR*.

```
// input parameters
uint8_t mode, f0, BW, Ns, Ntar, Rmax, MTI, Mth;

// results output array
float distance[5], SNR[5];
```

Distance variable will be define as *float*. This type can be as high as 3.4028235 E+38 and as low as -3.4028235 E+38 and it is saved as 32 bits (4 bytes) of information.

We define the baud rate of the serial port and below the value of each one of the input variables of the uRAD configuration function.

```
void setup() {
  Serial.begin(250000);  // serial port baud rate

  mode = 3;  // triangular mode
  f0 = 5;    // starting at 24.005 GHz
  BW = 240;  // using all the BW available
  Ns = 200;  // 200 samples
  Ntar = 2;  // 2 targets of interest
  Rmax = 100;// searching along the full distance range
  MTI = 0;   // MTI disable because we want information of static
and moving targets
  Mth = 0;   //  parameter  not  used  because  “movement”  is  not
requested
```

Now, we load the desired configuration and turn on uRAD.

```
  // load the configuration
  my_uRAD.loadConfiguration(mode, f0, BW, Ns, Ntar, Rmax, MTI, Mth);

  digitalWrite(6, HIGH);   // switch ON uRAD
  delay(100);   // recommended delay after powering ON uRAD
```

Next, we select the parameters we want to measure using the uRAD detection function.

```
void loop() {
  // target detection request
  my_uRAD.detection(distance, 0, SNR, 0, 0, 0);
```

Finally, we print on screen through the serial port the distance at which the different targets are detected. To do this, we will formulate the condition that the SNR of those tagets must be greater than 0 dB to be detected by uRAD.

```
    for (uint8_t i = 0; i < Ntar; i++) { // iterate between targets
      if (SNR[i] > 0){           // if target is reflective enough,
print its distance
        Serial.println(" ");
        Serial.print("### Target ");
        Serial.print(i+1);
        Serial.println(" ###");

        Serial.print("Distance: ");
        Serial.print(distance[i]);
        Serial.println(" m");
      }
    }
}
```

Graphical User Interface 10

Installing the GUI

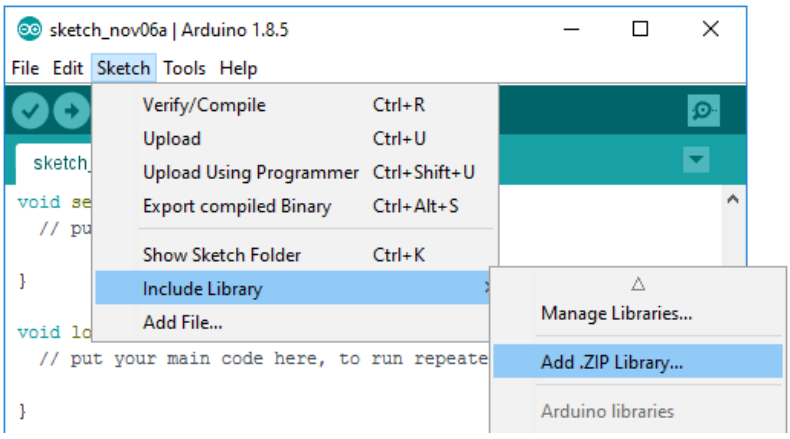
To facilitate the visualization of the results, as well as the configuration of uRAD, we have created a Graphical User Interface (GUI) that allows controlling uRAD in a simpler and more intuitive way than through the Arduino programs.

WARNING: The GUI is a program developed in Python that does not need any installation to run. However, before you can use the GUI, it is necessary to load a specific Arduino program on your board. In addition, it is necessary to install another different library for this program to work.

Follow these steps to install the new library and upload the program that allows you to use the GUI.

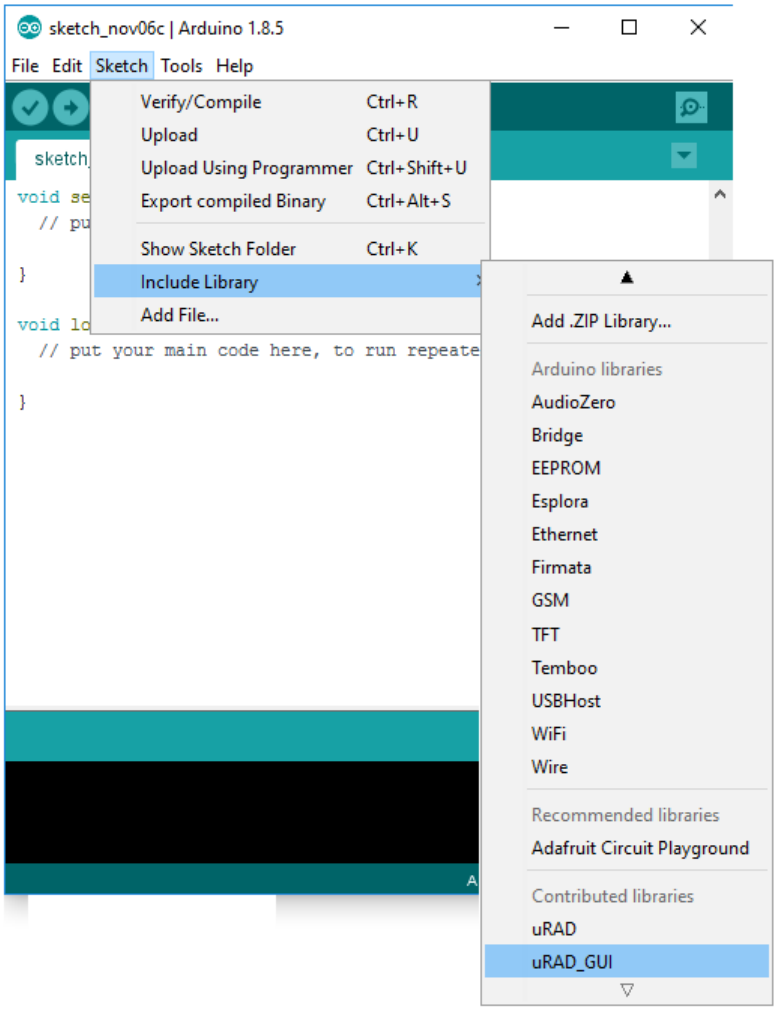
If you have not already done so, download from www.urad.es/en/mi-cuenta/downloads the folder ***Arduino_software.zip*** and save it in your hard disk. Unzip this folder and inside *Arduino_software\Library*, you will find the library ***uRAD_GUI.zip***.

In the same way we did in Chapter 4, to add the uRAD library to Arduino we will go to *Sketch > Include Library > Add .ZIP Library...*



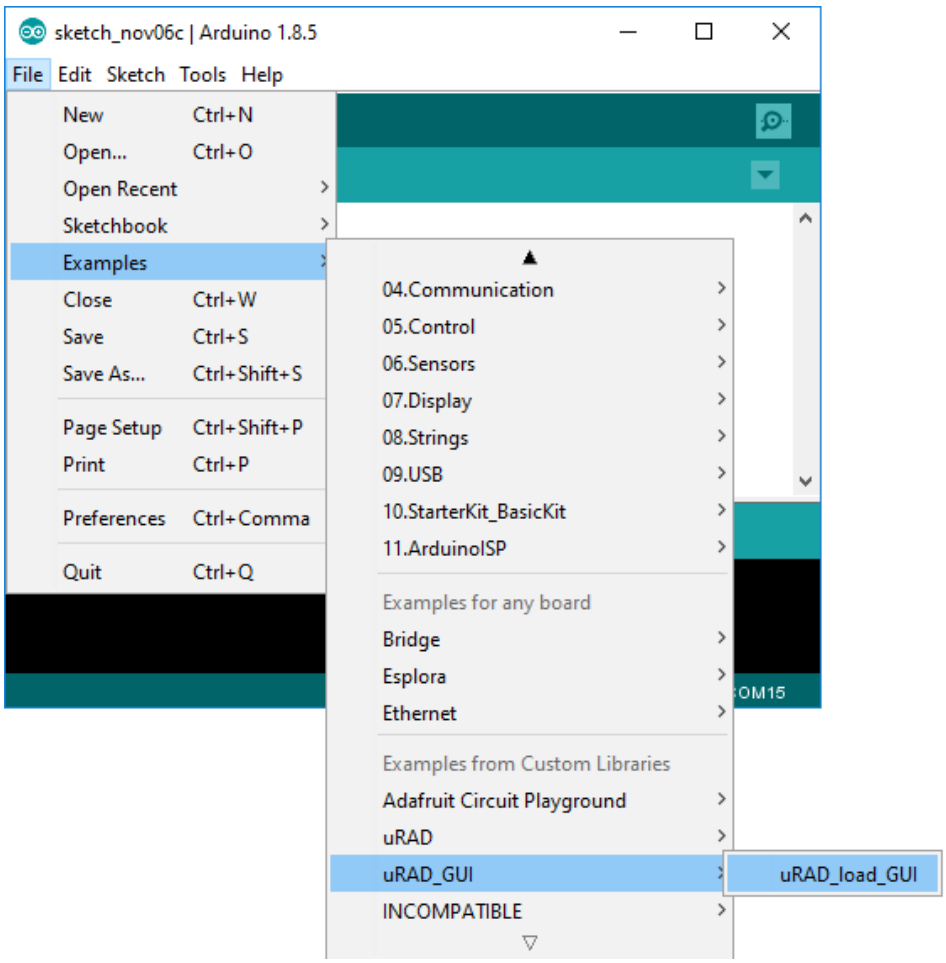
You will be asked to select the library that you want to add. To do this, go to the location where you have the file ***uRAD_GUI.zip*** and then select *Open*.

Now, go back to *Sketch > Include Library*. You should now see the library at the end of the drop-down menu.

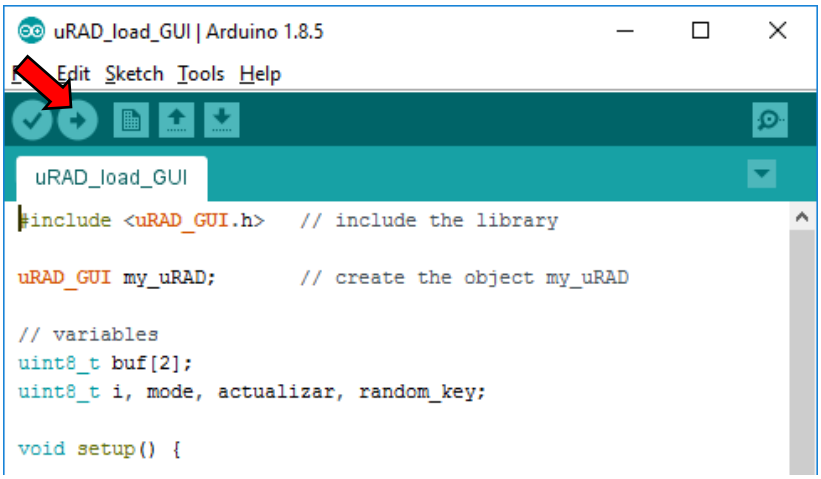


Once the library is installed, restart Arduino IDE so that you can see the example that you have to load on the card.

Navigate to the menu *File > Examples > uRAD_GUI > uRAD_load_GUI*. Making this, you open the program ***uRAD_load_GUI.ino***.



Upload this program to your Arduino card by clicking on the arrow in the upper right.

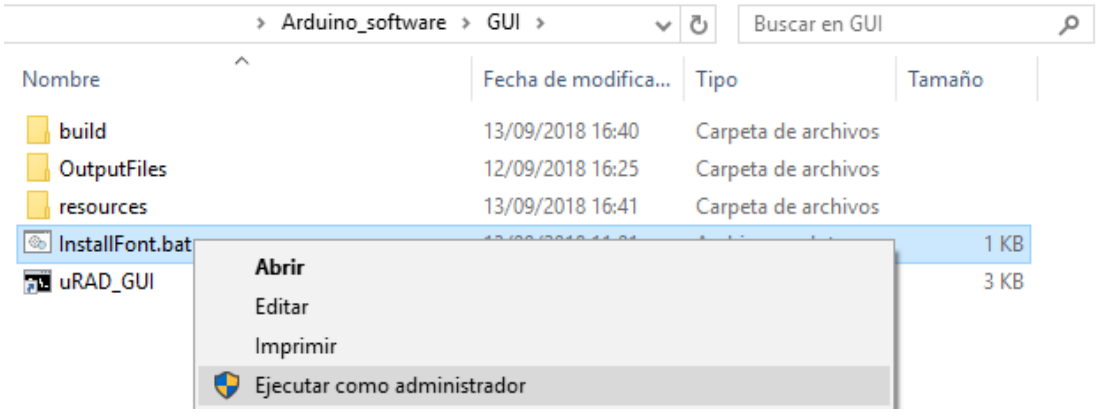


You can also find the program **uRAD.load_GUI.ino** in *Arduino_software\Library\uRAD_GUI\examples\uRAD_load_GUI*. By double clicking on the file, you will open the program in Arduino IDE in the same way.

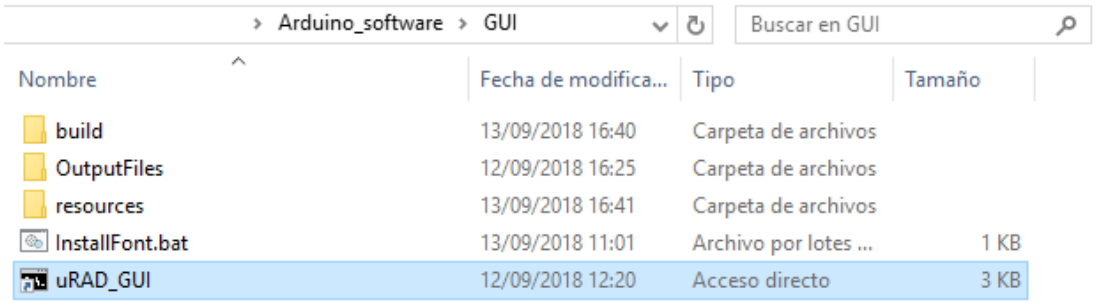
Depending on the operation system you are using, you have to follow different steps now:

On **Windows**:

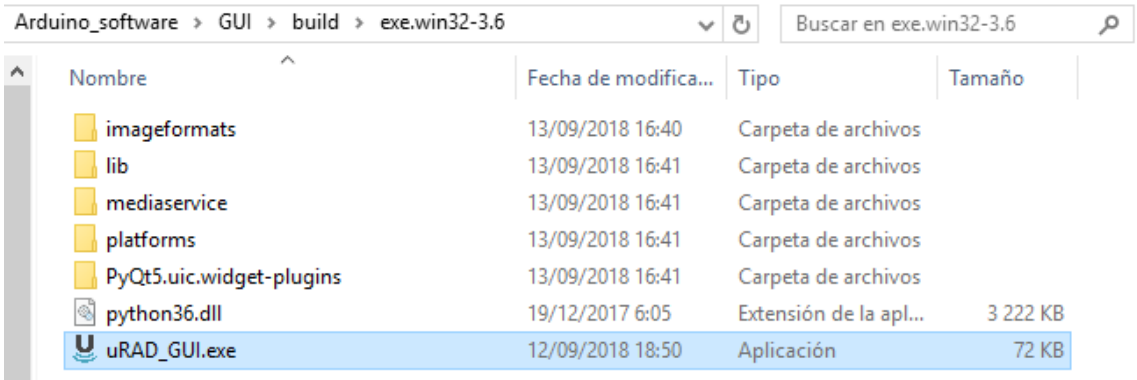
Navigate to *Arduino_software\GUI* and click with the right bottom on the file ***InstallFont.bat***. Select the option *Execute as administrator*. This step will install the font in the system necessary to correctly the GUI correctly.



Finally, once ***uRAD_load_GUI.ino*** is already uploaded in your board and you have installed the font, you just have to navigate to *Arduino_software\GUI* and execute the program ***uRAD_GUI*** to launch the GUI.



This file is like a shortcut of ***uRAD.GUI.exe*** that is located in *Arduino_software\GUI\build\ exe.win32-3.6*.



On **Linux**:

You must firs install Python 3 to run GUI, whether you have not installed it yet. Open a terminal and type the following commands:

```
sudo apt-get install python3
sudo apt-get install python3-pip
```

You must also install the necessary Python libraries. Type in the terminal:

```
sudo pip3 install pyqt5
sudo pip3 install numpy
sudo pip3 install pyqtgraph
sudo pip3 install pyserial
sudo pip3 install datetime
```

To launch the GUI, execute in Python the program ***uRAD_GUI.py*** located in the folder *Arduino_software_linux\GUI* or type in the terminal:

```
python3 uRAD_GUI.py
```

On **Mac**:

You must firs install Python 3 to run GUI, whether you have not installed it yet. Open a terminal and type the following commands:

```
brew install python3
```

You must also install the necessary Python libraries. Type in the terminal:

```
pip3 install pyqt5
pip3 install numpy
pip3 install pyqtgraph
pip3 install pyserial
pip3 install datetime
```

To launch the GUI, execute in Python the program ***uRAD_GUI.py*** located in the folder *Arduino_software_mac\GUI* or type in the terminal:

```
python3 uRAD_GUI.py
```

Selecting the Configuration Parameters

To launch the GUI, run the program ***uRAD_GUI***. After a few seconds, a window similar to the following will open.

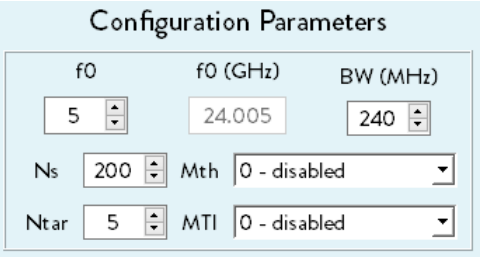
The GUI is divided mainly into two parts. The left part on a light blue background, corresponds to the configuration parameters, power buttons and saving the results. The largest part on the right contains the visualization of the results. We will now explain the options present in the configuration parameters part.

- **USB Port:** Below the uRAD symbol, there is the selection part of the USB port.

In the drop-down menu, you should select the USB port where your Arduino card is connected. If you have connected your card after launching the GUI, click *Refresh* to refresh the ports and see the USB port where you have it connected.

> *ADVICE:* You can know the USB port where your Arduino is connected, going to the Windows Device Manager.

- ***Configuration Parameters:*** The following image shows the part to enter the configuration parameters.

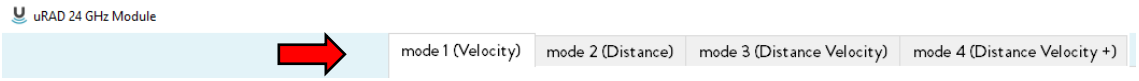


How you have already learned from Chapter 6 or from the User Manual, there 8 parameters that configure uRAD. In this box you can select 6 of them:

- f0: transmission frequency or starting ramp frequency (next to it is the real frequency value in GHz).
- BW: bandwidth in MHz (only selectable in modes 2, 3 and 4).
- Ns: number of samples.
- Ntar: number of targets.
- Mth: movement detection threshold.
- MTI: activating the mode Moving Target Indicator (discard static targets).

ADVICE: If you place the mouse a few seconds on the value of f0, BW, Ns or Ntar without clicking, a small help will appear showing the range of values you can select.

For the selection of the operating mode, you must click on the corresponding tab at the top of the results window.



On the other hand, the parameter corresponding to Rmax/Vmax is not available as such in the GUI. If you want to visualize the results up to a certain distance or speed, it can be done by adjusting the axes of the graphs, as we will see later.

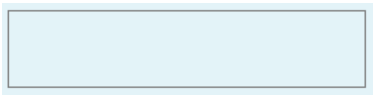
- ***Update Parameters* and *Run/Stop button*:** These two buttons are used to turn on and off uRAD and update the configuration.

Click on the *Run* button to turn on uRAD. Once running, click on *Stop* to turn off uRAD.

If you change some value in *Configuration Parameters*, the change will not be effective until you click on the *Update Parameters* button.

It is not necessary to stop uRAD to update the configuration, nor is it necessary to stop it to change the mode.

- **Error text box:** Below the *Run* button, there is a small empty box.



If an error occurs during the execution of the GUI, an error message will appear in this box.

- ***Save Results Data* checkbox:** In the lower part, there is a check box to save the results.

When you check this checkbox, the GUI starts saving automatically the distance, velocity and SNR results of the detected targets. A text file named ***results.txt*** is created in the folder *Arduino_software\Library\GUI\ OutputFiles*. When you uncheck the box, the program stops saving. This file is never overwritten, the program writes the results consecutively when you check and uncheck the box.

Every line in ***results.txt*** corresponds with one measurement y every measurement saves the information in 21 columns:

- Column 1 is the mode.
- Columns from 2 to 16 is distance, velocity and SNR of the 5 possible targets (0 indicates no result)
- Column 17 is the Mth value.
- Column 18 is 1 if movement = true or 0 if movement = false.
- Column 19 si the MTI value: 0 disable, 1 enable.
- Columns 20 and 21 are the date and time.

REMINDER: Depending on the mode, you will get results only of velocity and SNR (mode 1), only distance and SNR (mode 2), or velocity, distance and SNR (mode 3 and 4). If there is no result, the column will have a value equal to 0.

Even if only one target has been selected (Ntar = 1), the text file ***results.txt*** always has the same columns. The GUI saves a value equal to 0 in the columns that correspond to more objectives than the selected ones.

- ***Save IQ data* checkbox:** In the lower part, a box appears to save the RAW values of the I and Q components.

☐ Save IQ Data

When you check this box, several .txt files are created with the values of I and Q. The files that are created depend on the mode. The following table will serve as a guide to know what files are created in each mode.

Modo 1	
I_CW.txt	I signal values with Ns samples
Q_CW.txt	Q signal values with Ns samples
Modo 2	
I_FMCW_sawtooth.txt	I ramp values
Q_FMCW_sawtooth.txt	Q ramp values
Modo 3	
I_up_FMCW_triangle.txt	I values of the ascending ramp
Q_up_FMCW_triangle.txt	Q values of the ascending ramp
I_down_FMCW_triangle.txt	I values of the descending ramp
Q_down_FMCW_triangle.txt	Q values of the descending ramp
Modo 4	
I_up_1_FMCW_triangle.txt	I values of the first ascending ramp
Q_up_1_FMCW_triangle.txt	Q values of the first ascending ramp
I_down_1_FMCW_triangle.txt	I values of the first descending ramp
Q_down_1_FMCW_triangle.txt	Q values of the first descending ramp
I_up_2_FMCW_triangle.txt	I values of the second ascending ramp
Q_up_2_FMCW_triangle.txt	Q values of the second ascending ramp
I_down_2_FMCW_triangle.txt	I values of the second descending ramp
Q_down_2_FMCW_triangle.txt	Q values of the second descending ramp

Each line in these files is one measurement. Each line has so many columns as number of samples (Ns) are selected plus two additional columns with the date and time.

The exception is given in mode 4 in the files of the second ascending and descending ramp. Each measurement of those files has 0.75*Ns columns since that second ramp is shorter (plus two columns with the date and time).

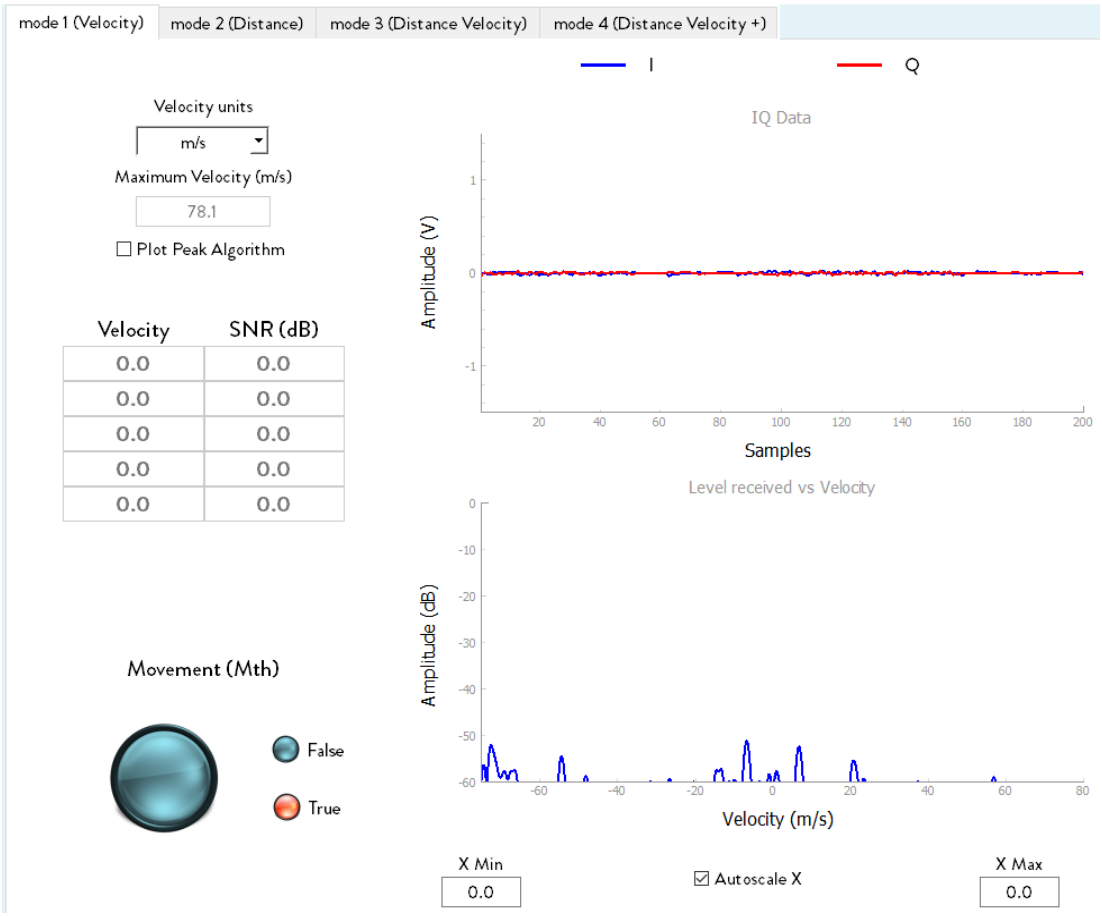
REMINDER: Depending on the mode, uRAD emits different signal with different ramps and therefore, the total received signal I and Q has different total length. For instance, with Ns = 100:

- if mode = 1 = 2 → length of I/Q = 100
- if mode = 3 → length of I/Q = 200 (100 + 100)
- if mode = 4 → length of I/Q = 350 (100 + 100 + 75 + 75)

When you uncheck the box, the program stops saving. These files are never overwritten, the program writes the results consecutively with their corresponding date and time.

Visualizing Results – Mode 1

Depending on the mode, the part of display the results changes slightly. In mode 1, the Doppler mode of only velocity, the appearance of the screen with the radar on is as follows:



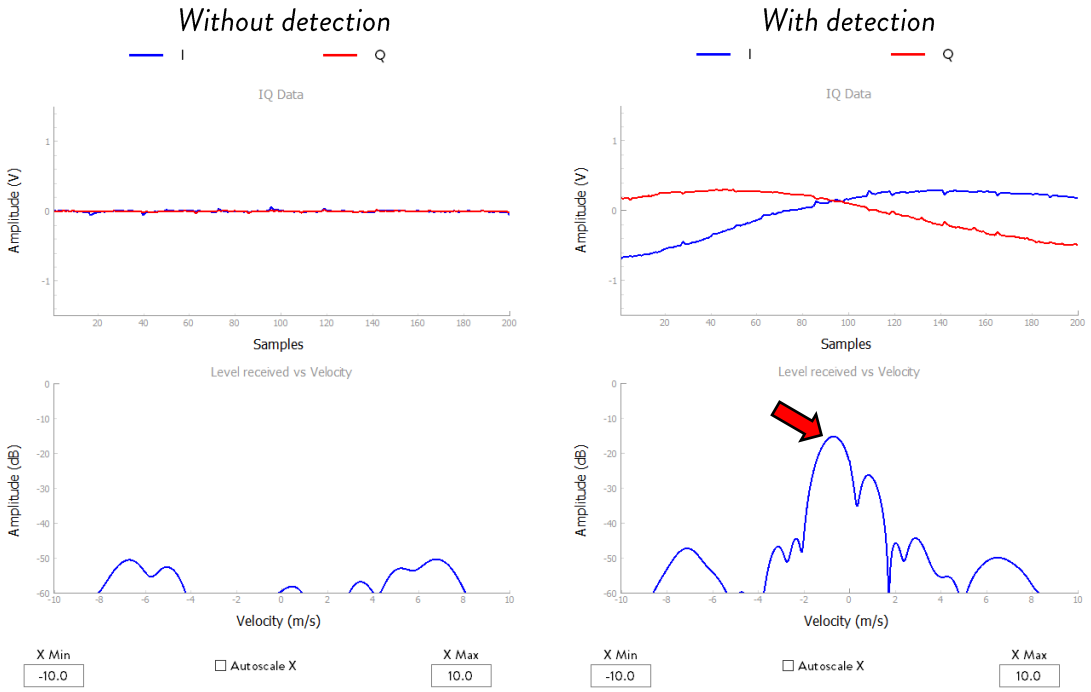
The upper right graph, titled *IQ Data*, show the I (blue) and Q (red) received signals. The x-axis corresponds with the number of samples, so whether you change *Ns* and click on

Update Parameters, the graph axis changes. The signal amplitude of I and Q is shown in volts, represented in the y-axis.

The I and Q signals are the phase and quadrature components of the received signal by uRAD. In other words, by building the complex signal I + *j*Q you would get the total signal received and from there, you can extract the magnitude and phase of the reflected wave.

The lower right graph, titled *Level received Vs Velocity*, corresponds to the fast Fourier transform (FFT) of the complex signal I + *j*Q. The FFT is of great interest because from it, you can obtain the information of the targets. The FFT, also called signal spectrum, shows the frequency components that comprise the total signal. Then, each of the relevant peaks that appear in the spectrum corresponds with a frequency, and therefore with a target detected by uRAD. By observing the position of that peak with respect to the x-axis, the velocity of the objective can be calculated. The FFT is represented in dB, therefore, the y-axis is the amplitude of the signal in dB, which is normalized to a maximum of 0 dB and a minimum of -60 dB.

We see below two situations where nothing is detected and where a target is detected at a certain speed, to see the differences in the graphs.



It can be clearly seen that when there is no detected target, no signal appears in *IQ data* or peak in the spectrum. When a target is detected, the variation of the I and Q signals is seen and a peak appears in the spectrum.

WARNING: On the spectrum, actually, many peaks are present. Some are targets, others correspond to noise and others may be the result of interference, bounces, etc. A peak detection algorithm has been programmed to identify those that really correspond to objectives.

To the left of the graphs, in the central part, a table appears where the results are shown. These results are extracted from the previous graphs.

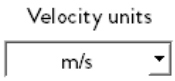
Velocity	SNR (dB)
-0.7	36.8
0.0	0.0
0.0	0.0

This table has as many rows as the number of targets, Ntar, you have selected. It shows the speed of the detected targets and their SNR in dB.

REMINDER: the SNR is the Signal to Noise Ratio. It is the difference in magnitude between the reflected signal due to the target and the noise background due to the whole system. It gives an idea of the amount of reflected signal from each target and, therefore, its size and reflectivity.

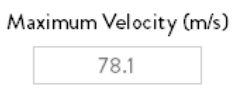
Negative velocities means that the target approaches uRAD while positive velocities means that the target is moving away from uRAD.

In the left-hand part above, you can select with the drop-down in which units the velocity results are displayed, *Velocity units*. You can select meters per second (m/s), kilometers per hour (km/h) and miles per hour (mph).



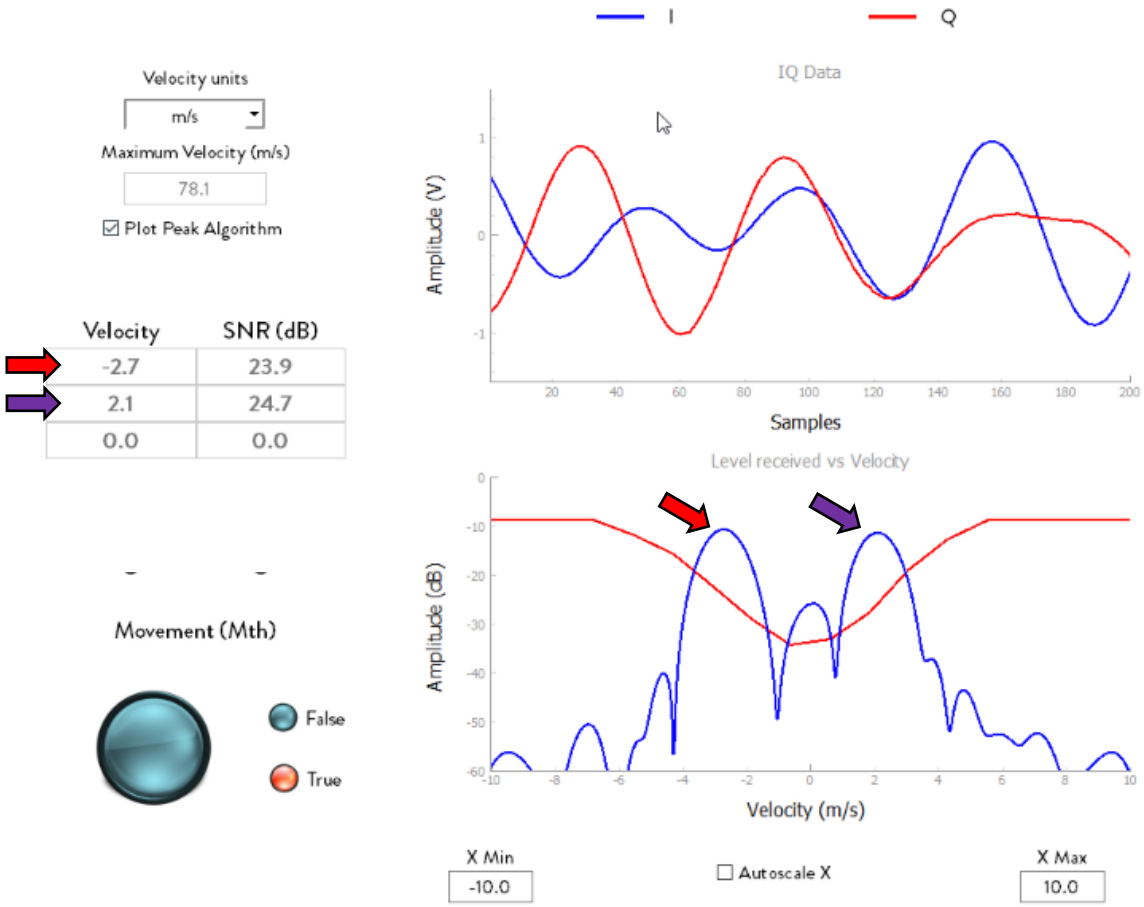
Below, an information box allows you to see the maximum velocity that can be measured. This maximum velocity depends only on the f0 in GHz selected:

$$Velocity_{max} = \frac{1875}{f0\ (GHz)}$$



Just below, there is a checkbox named *Plot Peak Algorithm*. When this checkbox is activated, a red line appears on the spectrum graph. This line represents the peak detection algorithm that we mentioned above. In order for a peak to be considered a detected target, it must at least exceed this red line, although it must also comply with some other condition that prevents it from being discarded.

We see below an example where two objectives are detected, one approaching and another moving away, its relationship of peaks in the spectrum and the results in the table.



Por último, en la parte inferior izquierda, se encuentra el led que te avisa si se ha detectado un movimiento.

For this led to work, you must first activate it in *Configuration Parameters*, in the drop-down of the *Mth* parameter. The led will turn red if there is a movement detected and blue if there is not.

Visualizing Results – Mode 2

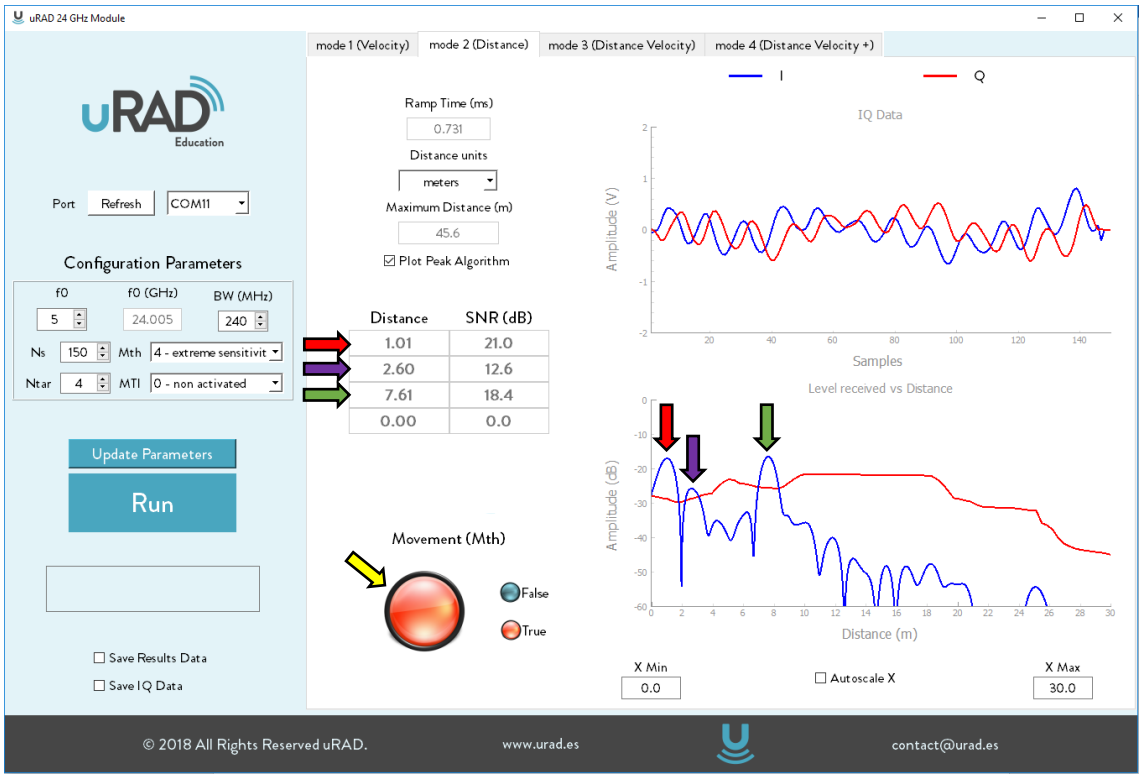
The results shown in mode 2 are similar to those in mode 1, except that in this mode only distance results are shown, instead of velocity.

The spectrum graph is titled in this case *Level recieved Vs Distance*. Because in this case the signal emitted is modulated in frequency by means of a ramp that varies in time, the x-axis of the spectrum corresponds to the distance of the targets to uRAD.

In the upper left part, the information of the ramp duration in milliseconds is shown, *Ramp Time* and the maximum detected distance, *Maximum Distance*. You can also select the *Distance units*. The maximum distance is a function of the configuration parameters according to the following formula:

$$Distance_{max} = 75 \times \frac{Ns}{BW}$$

We see below an example where 3 targets have been detected at 3 different distances and, in addition, one or several of them are in motion, since the movement LED is activated.

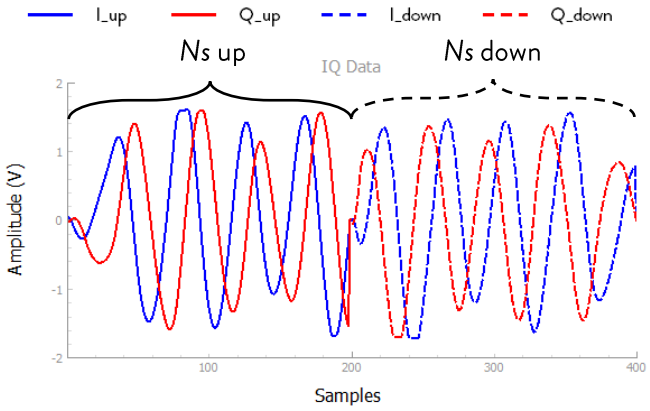


Visualizing Results – Mode 3

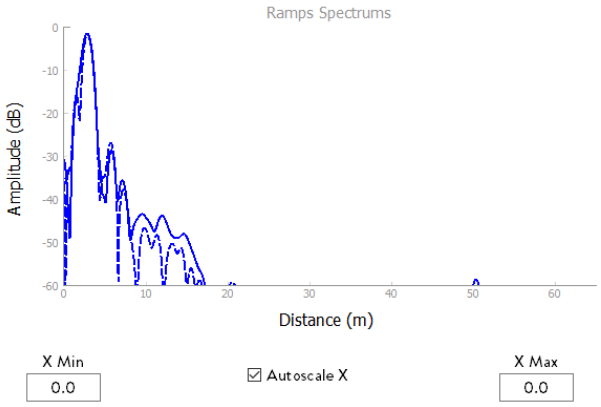
Mode 3 allows you to calculate both the velocity and distance of the targets because the signal emitted varies in frequency according to a triangular shape, with a ramp up and down.

ADVICE: If you want to learn a little more about how different frequency ramps allow different results, we recommend reading the uRAD White Paper, available on www.urad.es.

The *IQ Data* graph, shows in this case the I and Q values for the up and down ramp. The signal received during the up ramp is drawn with a continuous line and the one received during the down ramp with dashed line. Therefore, the x-axis has twice as many samples as the selected Ns, since they are Ns for the up and Ns for the down. Let us see an example with Ns = 200, the maximum number of samples.

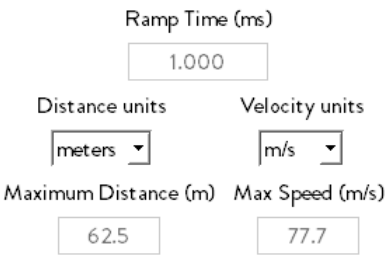


In the spectrum graph, two signals also appear. In the same way, the continuous line corresponds to the FFT of the complex signal I + *j*Q of the up ramp, and the discontinuous one with the FFT of the signal of the down ramp.



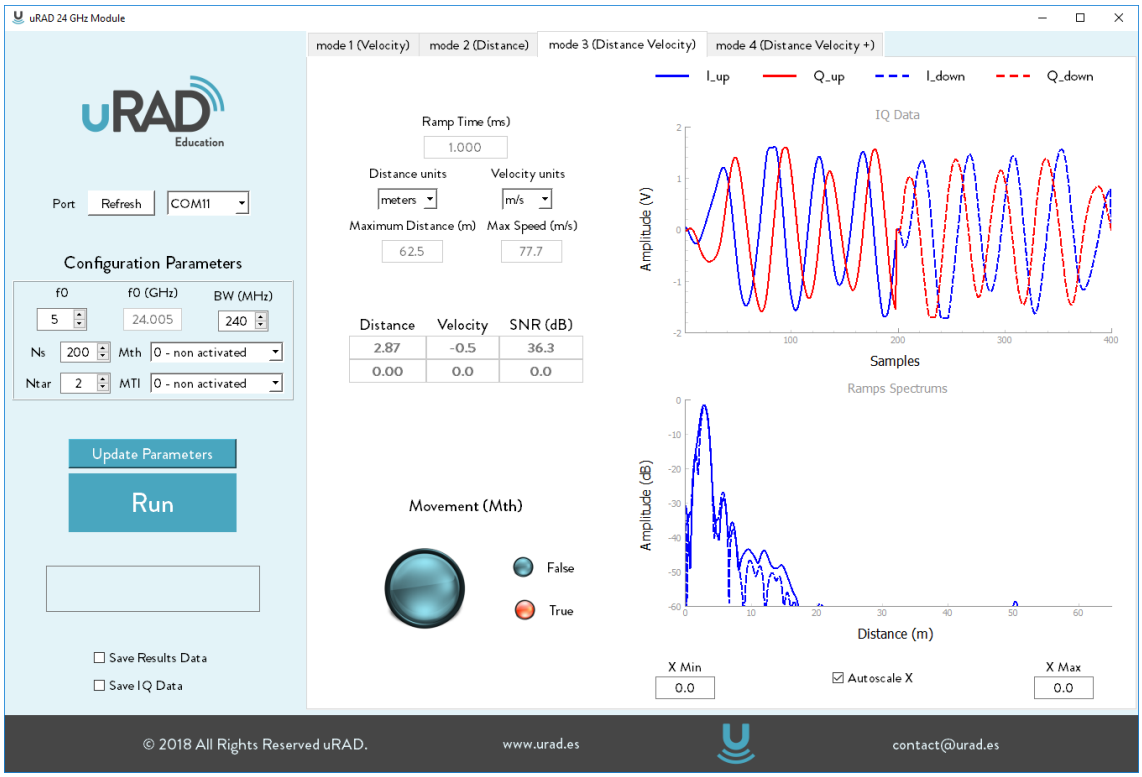
En este modo no existe la opción de pintar el algoritmo de detección de pico ya que hemos considerado que pintar dos líneas más sobre está gráfica sería poco claro.

In this mode, you can also select the units of both measured distance and velocity, as well as visualize information of ramp time, maximum velocity and distance. Therefore, the results table shows the distance, velocity and SNR information of the detected targets.



Distance	Velocity	SNR (dB)
2.87	-0.5	36.3
0.00	0.0	0.0

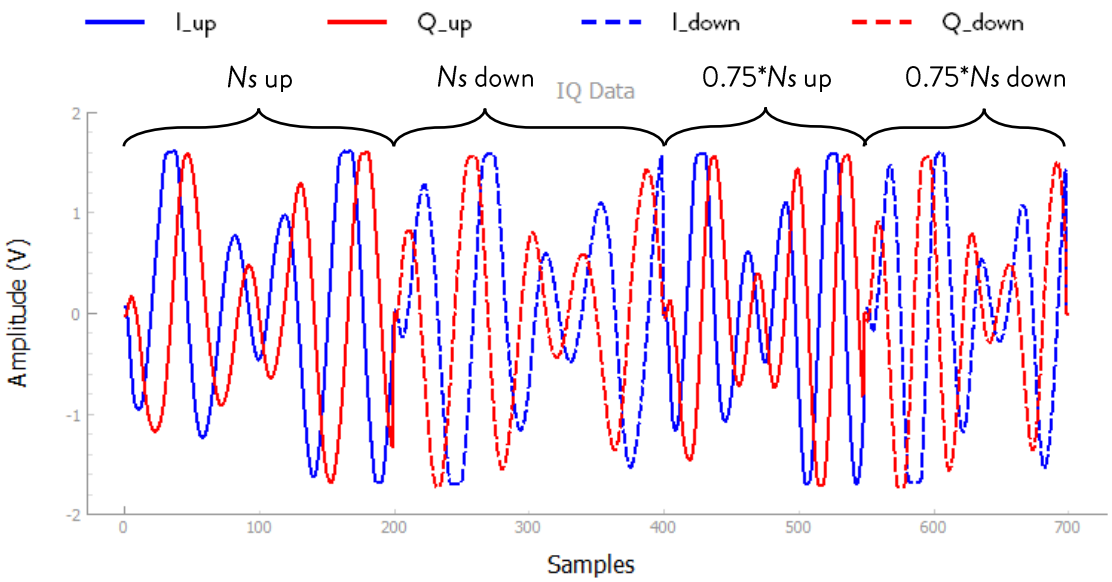
We see below a typical example, where a single target has been detected at 2.87 meters and approaching at velocity of 0.5 m/s.



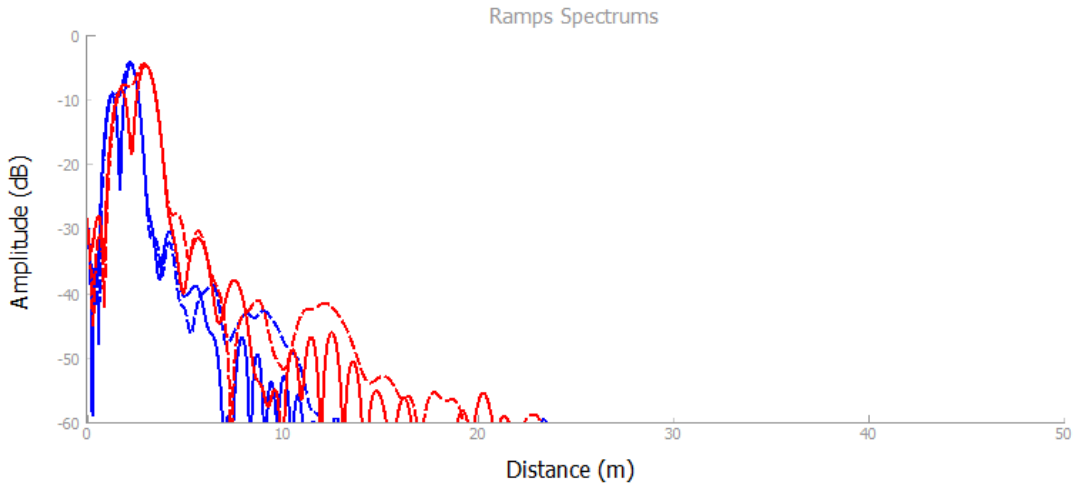
Visualizing Resuls – Mode 4

Mode 4 is very similar to mode 3, since it also allows obtaining velocity and distance information. In this mode, the transmitted signal also varies in frequency according to two consecutive triangular signals of different duration. The first triangular is formed by an up ramp and another down ramp, of Ns samples each one; and the second triangular by an up and down ramp of 0.75 * Ns samples each one.

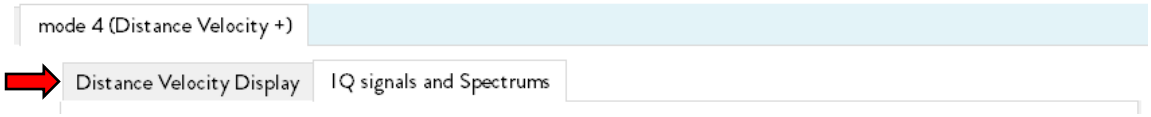
This can be easily seen in the *IQ Data* graphic. Let us see an example of a signal received with Ns = 200 samples.



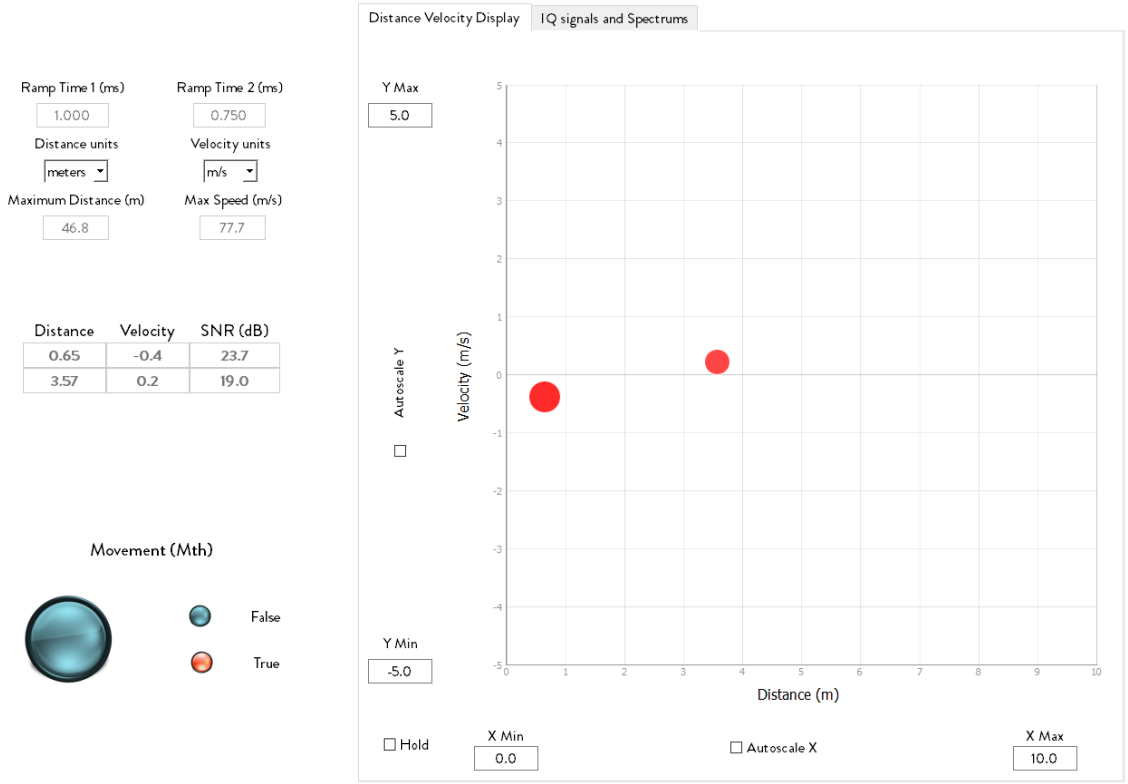
Consequently, the spectrum will also show 4 different signals. The blue continuous line corresponds with the FFT of the first up ramp, the blue discontinuous one, with the first down ramp, the red continuous one, with the second up ramp and the red discontinuous one with the second down ramp.



In mode 4, we have created an additional window to visualize the distance and velocity results of the targets in a graphical form. You can select it in the tab *Distance Velocity Display*.



Clicking on it, you will see a screen similar to this one. Each detected target is represented by a red dot. The x-axis corresponds to the distance at which the objective is located and the y-axis represents the velocity. The size and opacity of the objective is directly proportional to its SNR. Targets with high SNR will look bigger and less transparent.

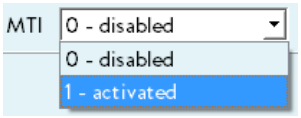


Down to the left, there is a checkbox with the word *hold*. When activated, the results painted in this graph do not disappear until after a few seconds, being easier to visualize the path of the objectives.

The rest of options and information of mode 4 is equal to mode 3, except that the information of duration of the ramps of the second triangular signal is also presented.

MTI Mode

The MTI mode, selectable in the part of *Configuration Parameters*, allows us to discard all those targets that are static. The drop-down menu is presented as follows:



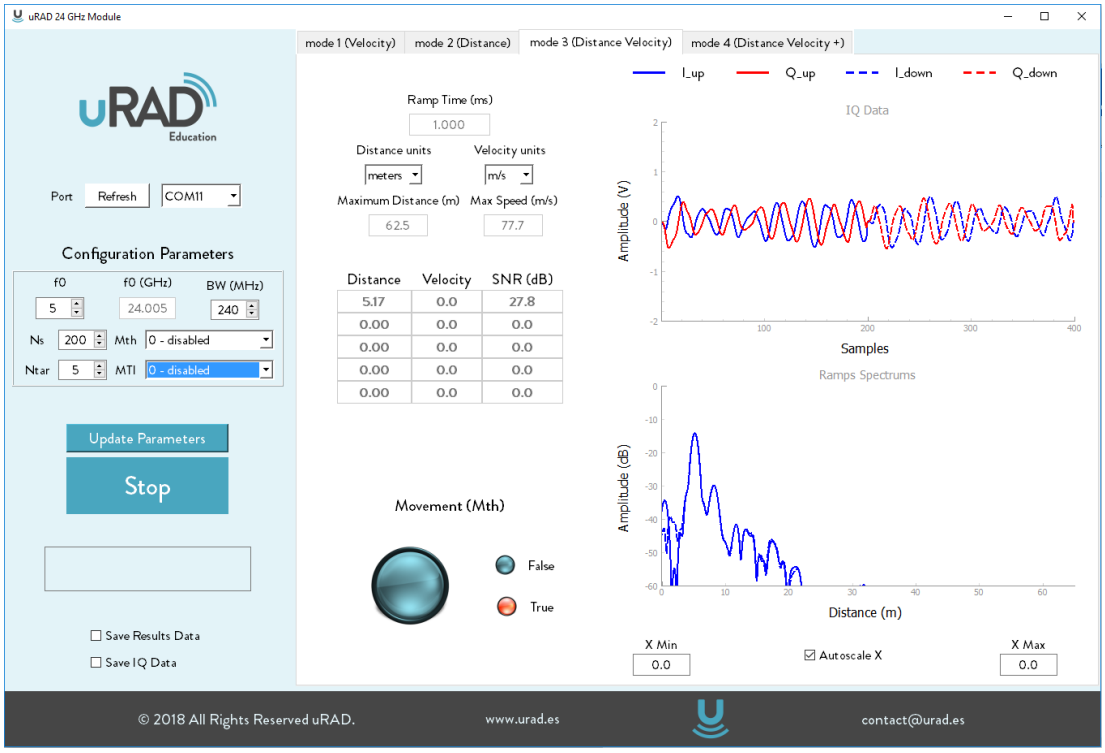
This mode only makes sense in modes 2, 3 and 4, since mode 1 is only for detecting the velocity of moving targets and then, by default, static targets are not detected.

We see below an example of using the MTI mode. In this scenario, there is a wall at 5 meters from uRAD, which remains static pointing to the wall.

With the MTI mode disabled, the signal reflected in the *IQ Data* graph is perfectly visible and the peak in the spectrum appears. The results table also shows the distance up to the wall.

On the second capture, the MTI mode has been activated. In this case, the signals I and Q have disappeared and the spectrum does not show any peak, so the wall is no longer detected and does not appear in the results table.

MTI mode disabled



MTI mode enabled