Arduino - 3G Module, Ultrasound, Keyboard & LED RGB

In this project we will learn to use a portion of all the options offered by the new module 3G/GPRS of cooking hacks, in this tutorial we will use it to play mp3 from a speaker connected directly to the module itself, to send a sms and mms, to take pictu

Written By: Feitan
INTRODUCTION

Welcome to new Arduino tutorial!

Take this opportunity to say that we are in big changes, both image and infrastructure to improve the quality of the project, I hope not to create too many inconveniences, also commenting that this closed May 1 Arduino Forum Academy (now Arduteka) for provide continuity on the official forum where solventaré Arduino.cc all doubts will arise from this tutorial in this post.

A new feature from now we will include a Premium Pack downloadable tutorial we make each and to get it, you only have to give us a Tweet or Post on Facebook! Budget, no?

Best of all, this pack brings some surprises! ;) In advance, I say inside will have Arduino files for each project libraries used in this tutorial, also sounds a version of the entire tutorial PDF to enjoy it offline .. and some more surprise! Download it now by clicking this button!
**Part 1: What are we going to do?**

- In this project we will learn to use a portion of all the options offered by the new **module 3G/GPRS of cooking hacks**, in this tutorial we will use it to play mp3 from a speaker connected directly to the module itself, to send a sms and mms, to take pictures and store them on an SD card, in addition to this, it will look like a $4 \times 4$ matrix keyboard is integrated into our projects Arduino, an RGB LED and an ultrasonic sensor, not bad, not? Now let's see what we do with all this!

- Let's build us a totally home alarm, through the ultrasonic sensor continuously scan them the space in front of you with a radius of $30^\circ$ approximately, when an object or person is placed in its field within a distance at which we establish, will sound an alarm, take a picture, the RGB LED that was previously green to blue...
and will give you 10 seconds to power off alarm through our matrix keyboard, if the disable will return again to scan the field, but if not! Played a strong sound
and will have to send an sms to our mobile phone and photograph to our email.

- Let's see it on video!

- I also remember that the official thread of this project, where you can evolve it, and ask questions of others will be THIS LINK hosted on the official website of Arduino.
Step 2

- Part 2: Designing the Hardware: Let’s start by knowing a little more about the new 3G module Cooking Hacks and that will be a fundamental part of this project.

- This module is the most complete that I could try to date as a single shield combines the following features: Support WCDMA and HSPA. Make and receive calls and video calls. Send sms and mms. Internal GPS support A-GPS (Assisted GPS) and S-GPS (GPS simultaneously).

- Video camera with photo and video support. We can connect a microphone, a speaker and a headset to handsfree. SD cards up to 32 GB We can use it as a 3G modem. Direct communication by HTTP / HTTPS. Upload and download files via FTP / FTPS. Send and receive emails POP3/SMTP. Play and record audio files.
Step 3

- This described what we can manage by AT commands from our Arduino as discussed in the programming part. As for specific connections to this module, we will use the output of "loudspeaker", the socket for the SIM card, microSD card socket and VGA camera.
Step 4

- We must also take into account the position of the two jumpers to switch the direction of communication, then we can control it from our Arduino, or directly by USB port using CuteCom, hyperterminal or similar.

- In the picture we can see all the available connections of the module.

- The good thing about this type of shields, is that being designed for Arduino, go directly "tapped" on it and will continue using almost all features of this because you just need three pins for operation, pin 0 and 1 for communication and pin 2 which enables or disables the module, something very interesting if we want to save power in our projects, we could be collecting data throughout the day, and only at a certain time, turns the module, we send the data, and off again! Now continue with the other components of our project!

- Intrusion detector, we will use a sensor ultrasonic ranging from 3cm to 300cm, as shown in the picture, only have three pins, two for power (Vcc and Gnd) and a third connected to our Arduino by which "shoot" the pulses and in turn receive, these pulses are just a beep at a frequency imperceptible to the human ear.
Step 5

- The operating sequence is as follows: If we send a pulse (HIGH) from Arduino from 2µS and 10µS, this will interpret it as a trigger (trigger) and emit a train of 8 beats at a frequency of 40Khz.

- Automatically we put the pin where we connected as input SIGNAL, because if that train of acoustic pulses bounce off an object and returns to the sensor, this will be detected and transformed into an electrical signal which we interpret from Arduino, similar to when ECO shout! and second we hear back.

- Depending on the time that it took the "ECO" to return to the sensor, we can determine the distance that has bounced, because we know the speed of propagation of sound in air, and the time taken to return to us, We will use this formula:

\[ d = \frac{1}{2} \times vt \]

- Where \( d \) is the distance that bounced in meters, \( V \) the velocity of propagation of sound in air (343m/s) and \( t \) the time taken in seconds, why not split between two is that the sound has made a round trip to the obstacle and a return to the sensor.
Now we just need to adapt this formula units as needed, for \( t \) we will get in microseconds, so instead of working with \( V \) in m / s we will transform in microseconds / cm, giving as approximate result 29\( \mu \)S / cm (1000000\( \mu \)S/34300cm) in this way, as discussed in the programming part, left us a very clean way to calculate the measure: distance in cm = time in microseconds / 29/2
Step 6

- Then we study the matrix keyboard with which we can disconnect our alarm using a key as long as you want.

- The operation is very simple, as its name suggests, is a matrix of connections arranged in rows and columns, from left to right and top to bottom, so that if you press the number 1 will be joining with row 1 column 1 if you press the number 0, we will be joining with row 4 column 2, etc. .. better see a representation of its internal connections

- See Image 2

- As for the pin configuration is very simple, with the keys to us, we will have from left to right, first by rows and then columns.

- See Image 3

- To manage these combinations have the Seller keypad.h we will greatly facilitate programming as we shall see later.
Finally, let's use an RGB LED of common anode to display the different states of our alarm when you are expecting an intruder, be green, during the time frame to clear, will be in blue, and if we exhaust the time, or introduce a wrong password, it will be red.

This LED diode is itself three integrated into a single package diodes, one of Red (color R), another, Green (G) and a color Blue (B) and has the characteristic that can theoretically reproduce any color combining the intensities of each.

As noted, this LED RGB in particular, is common anode, ie, no one blade constantly connected to positive, and to polarize each color, we take a negative with a corresponding resistance to limit the current passing through it by reference to the longest, which will correspond to the anode.
Finally, if we know the combination of intensities of each color for one in particular, we can use any program of photographic design to remove a color palette, and on it, selecting a color, we see that returns a decimal value R, G and B from 0 to 255, which coincidentally corresponds to our range of pwm.

With this I hope I have clarified the whole issue of hardware for this project, so I am going to just list that must be connected to each pin of Arduino to correspond with our program.

- *Pin 3: Row 1 keyboard.
- *Pin 4: Rank 2 keyboard
- *Pin 5: Row 3 keyboard
- *Pin 6: Rank 4 keyboard
- *Pin 7: SIGNAL sensor ultrasound
- *Pin 9: Red color RGB Led
Step 9

- Pin 10: Red color RGB Led
- Pin 11: Red color RGB Led
- Pin 14: Column 1 keyboard
- Pin 15: Column 1 keyboard
- Pin 16: Column 1 keyboard
- Pin 17: Column 1 keyboard

As we recall, the analog inputs can be used as digital inputs or outputs in addition to its function analog / digital converter.
Step 10

- Programming

- When you abráis the downloaded file, you will see that it is divided into different tabs, so we will analyze one by one.

- Consider the code from the first tab, where have all the configuration and program loop, I remind you that that code will detail that we have not seen so far, if you have any questions, I recommend reviewing the previous tutorials where you will find all the information.

- [Code Here]

- If you start reading the code, once past the Seller we will include statements of the variables that we use and we find something new, shaping our matrix keyboard, as we will see, using this Seller keypad. h can not be much easier to configure!

- [Code Here]
Step 11

- The first thing you have to do is tell you how many rows (rows) and columns (cols) will have, then create an array like our keyboard, we do this by creating a two-dimensional array as you can see,

- where you have to change the function of the keyboard characters you who are using now we say the pins assigned to the rows and columns and created an "object" Keypad with the following syntax:

  Keypad (makeKeymap (userKeymap), row [], col [], rows, cols)

- We have it set up for use in our Arduino project!

- Then we started with the setup.

- Note that you can activate the serial communication with a baud rate of 115200, so, that is the rate at which work the 3G module.

- Serial.begin(115200);
Step 12

ACTIVATE 3G MODULE

To activate the 3G module, we must send a pulse of 2sg by pin number 2 if quisiéremos off, should do the same, this will be very useful for Arduino projects that are battery powered and need not be constantly active module.

- Serial.printLn("Activando modulo 3G...");
- digitalWrite(onModulePin,HIGH);
- delay(2000);
- digitalWrite(onModulePin,LOW);
- After a timeout elapsed before the device starts successfully, start with your configuration by AT commands.
- The AT commands were developed in 1977 by Dennis Hayes as a communication interface with a modem in order to configure and provide instructions as call, hang up, etc ...
- Over time were Microcomm and U.S. Robotics companies which continued to develop and expand the package instructions to universalize.
These commands and the call by the abbreviation ATention.

Although initially the AT commands are designed to work with modems, have been adopted as standard for communication with mobile terminals.

Thus, all mobile phones have a specific set of AT commands that interfaces to configure and provide instructions to terminals.

So each mobile terminal has a set of AT commands determined that lets you perform all the tasks for which it was designed, in our case, from checking our position to open a 3G data connection.

So let's start to configure!

The way we have to send AT commands to your module, is sending him text strings via serial communication via Serial.println instruction and derivatives.
First we are going to send AT most basic command, which is simply "AT" with this we will ensure that communication with the module, because each time we send a command to the module, and this receives and succeeds, it returns an OK, hence after each module will wait the "OK" to continue with the program, we have done this by saying that while statement known to not receive a K, not follow forward.

- Serial.println("AT");
- while(Serial.read()!='K');
- Serial.println("Existe comunicacion...");
- If you want to have all the reference guide AT commands in this module, you can download it from this link:
- To avoid duplicating all the code, I leave a small summary of the AT commands used in this setup:
- AT + CPIN = 0000 Insert the pin of our SIM card.
Step 15

**AT COMMANDS**

- **AT + CMMSCURL = "mmss"** URL of our operator MMS server without http:// in front. (We put the backslash to not take us to the quotes as string, since the url must be in quotes).

- **AT + CMMSPROTO = 1, "193209134141", 80** establish the parameters of the protocol (0 = 1 = http wap), the MMS proxy and port of our operator.

- **AT + CGSOCKCONT = 1, "IP", "mms"** parameters define PDP (Packet Data Protocol) among which is the access point (APN) MMS for our operator, in our case, "mms"

- **FSLOCA AT + = 1** Select the SD as storage (0 = 1 = C:/ D:/).

- With these parameters set we will have our 3G module ready to start running the loop!
Step 16

- Consider a flow chart to understand quickly. (See Image 1)

- When the distance is less than that set, store it in a variable that the alarm is on ...
  
  \[
  \text{alarma} = \text{true;}
  \]

- We found that the sms sent counter has not reached its limit ...
  
  \[
  \text{if (count < ; maxims)}
  \]

- We call the function we lit blue color LED RGB Led ...
  
  \[
  \text{ledAzul()};
  \]
• We reproduce an mp3 and take a picture, make a note that the duration delay our mp3 to prevent further executing instructions during that time and one of 500 milliseconds to ensure that the picture was stored properly.

  ```
  sonidoAlarma();
  delay(1000);
  tomaFoto();
  delay(500);
  ```

• Finally we store the time it takes connected plate to use as a counter and then compare the time for disconnection.

• The cycle while last as long as we set the timeDesc variable because it will continuously check that the difference between time2 (while constantly updated) and time1 (time is set to activate the alarm) is less than timeDesc.
Step 18

- Within this cycle we will find the statement to read when we pressed a key to our keypad:

```c
char customKey =
customKeypad.getKey();
```

- That is, constantly to be stored in customKey the character that we hold, thereupon put a condition such that if the value stored in customKey is different NO_KEY (no key pressed) we write it to a buffer to compare then with our default password

```c
if (customKey != NO_KEY)
{buffer[x]=customKey; x++;}
```

- So we will fill up the buffer that has the defined length.

- When you have reached that point, just compare strings stored in buffer and key to make the decision to place the alarm variable false or true depending on whether or not disabled.
Step 19

- To complete the loop, if the alarm condition is equal to true, execute the functions that put us in the red led, reproduce us a stronger sound alarm, it will send a sms and finally a mms.

- Now let's all call functions that separately.

- First, we see the functions that we change our color LED RGB Led, as you can see there is no mystery, just put in LOW that we pin that corresponds to the color you want to show because we do not have to mix any color for those who need it.

- Code

- Now for the function that we will measure the distance sends us our ultrasonic sensor connected to Arduino, as I explained in the hardware section,

- we will launch a pulse for 5 milliseconds pingPin pin to fire the train of high frequency pulses, and automatically we'll get to hear, once received echo, store it and transform it into the distance through the formula explained.

- Code
Step 20

To play sounds we will perform two functions, one for each mp3 stored then need to call the specific file, as we will see are the same, only changing the name of the file to play.

The AT commands that we used are:

- AT +D :/ Audio FSCD (We say it is located in the Audio folder of our SD card, which is where mp3 are.)
- AT = 3 + CSDVC (We tell the Speaker that use to play the mp3 (Handsfree 1 =, 2 = speaker, loudspeaker 3 =, 4 = interface pcm))
- AT + CCMXPLAY = "alarma.MP3" 0 (We tell the name of the file to play and if we play it during a call or not.)

To take the photo, we see that after setting up the camera, and the photo is taken, we must give to save, but this time instead of returning an OK, we will return the file name of the photo that we save in an array to later use in sending the mms.

Code
Step 21

AT COMMANDS

- The AT commands that we used are:
  - AT + CCAMs (started the camera, and assures us that it is properly connected.)
  - AT = 640,480 + CCAMSETD (We adjust the picture resolution.)
  - AT + CCAMTP (take the picture.)
  - AT + CCAMEP (We keep photography.)
  - AT + CCAME (stopped the camera.)

- To send sms in addition to the AT command normal, the characteristics of these modules tells us to give to send the sms one Ctrl + Z and CR + LF, this will do the instruction Serial.Write(),

- which as remember, we used to send binary data via serial port, instead of the characters directly, to send it to Ctrl + Z 0x1A, 0x0D 0x0A for CR and LF.
Step 22

- **Code**

- **AT COMMANDS**

- The AT commands that we used are:
  - AT + CMGF = 1 (selects the format of the sms (0 = PDU 1 = Text))
  - AT + CMGS = "phone" (Phone is the variable where we stored our target.)
  - And finally, prepare and send to our email our mms or mobile phone with picture of intruder alarm has sounded.

- **Code**
Step 23

AT COMMANDS

- AT commands we have used have been:
- FSCD = AT + D: (We indicate the drive where our picture is stored.)
- FSCD = AT + PICTURE (We tell the folder where it is stored.)
- AT = 1 + CMMSEDIT (went into editing mode mms.)
- AT + CMMSDOWN = "TITLE", 6 (We say that the title of mms is 6 letters. (Then you have to give the title to save it automatically.))
- AT + CMMSDOWN = "FILE", 4, "name" (We tell him what is going to send a file with the name stored in the array name)
- AT + CMMSRECP = "email" (We add as recipient an email or a phone number.)
- AT + CMMSSEND (We ship!)
Step 24

- That's it!

- All doubts of the tutorial will be able to comment on this link (arduino.cc/forum/index.php/topic,105734.0.html)