ThinkFreedom Hardware Stack

Manual for the Prototype



### Contents

### 

1. **Materials**
2. **Setting up Xbee**
3. **Assembling Controller**
4. **Assembling Control Unit**
5. **How It Works**
6. **Further Upgrades**

**1.Materials**

1x Arduino Uno

1x Arduino micro

1x Raspberry Pi (model doesn’t really matter but it would be better to get a Raspberry 2)

1x USB micro for power

1x micro SD card at least 4GB for Pi 2 -or- 1x SD card at least 4GB for Pi 1

1x SD card Adapter for accesing to the SD card or micro SD card

1x HDMI

2x Xbee S1

2x Xbee breakout board

2x Breadboard (you can do it with one but it will be more helpful with two)

2x USB female connector type A

1x USB cable male to male type A

1x USB male A to B (for programming the arduino)

1X Button and 1X 10kΩ resistor (depending on the number of inputs you want. Every input takes one button and one resistor)

1x LED

1X 330 Ohms resistor

Some Jumper wires

Some Pins (male and female)

9x DIP switches , including 2 additional switches on each one. Each DIP switch has 4 pins .Pins 1 and 2 are closer to the OFF state counting from the left to the right and Pins 3 and 4 are in the ON state counting from the left to the right .

And for later if you want to solder everything together :

1x Soldering board

2x Prototyping Box

(Everything can be found on ebay under 10$ (except the Raspberry Pi and the XBees )).

**2.Setting up the Xbee**

Things we are going to need here:

**Software**

Arduino IDE (<https://www.arduino.cc/en/main/software>). Select the one that matches your OS. This software will be used to program the arduino.

XCTU(<http://www.digi.com/support/productdetail?pid=3352>). Select the one that matches your OS. This software will be used to set up the XBee.

**Hardware**

Arduino Uno

Jumper Wires

USB Cable A to B

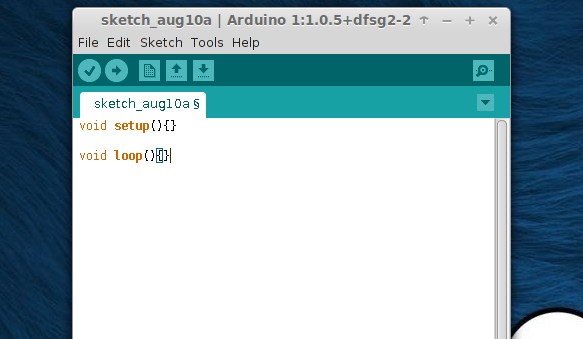
2x Xbee

2x Xbee breakout board

PC

To set up the Xbee we did it through an arduino board. It is more easy if buy a Xbee explorer USB but this way the budget for the project will rise. To set up the Xbee via an Arduino board there are some things to be done.

First, before you connect anything you upload a blank code to the Arduino as appears below. and upload it to the Arduino as USB asp from Tools-->Programmer.



Next connect the pin as appears below:

Xbee Arduino

+3.3v 3.3V

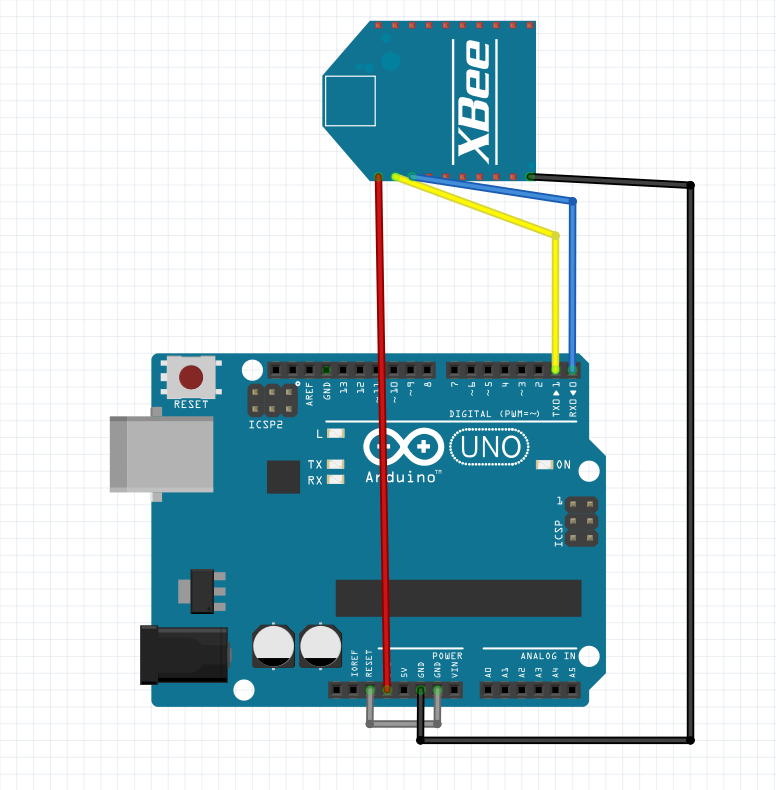
GND GND

TX TX

RX RX

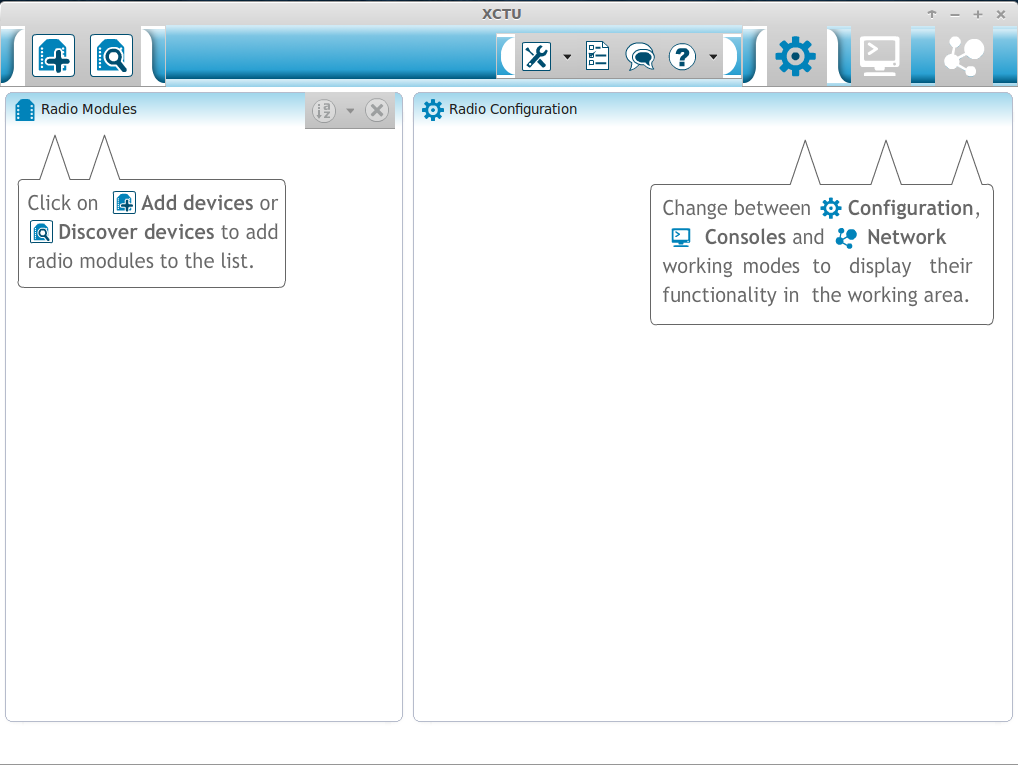
Also you can connect the reset and GRD of the Arduino or remove the ATmega chip. This way you can bypass the chip and set up the Xbee. This step is optional ,as the blank code you uploaded does the same thing.

The schematics for the connections is:



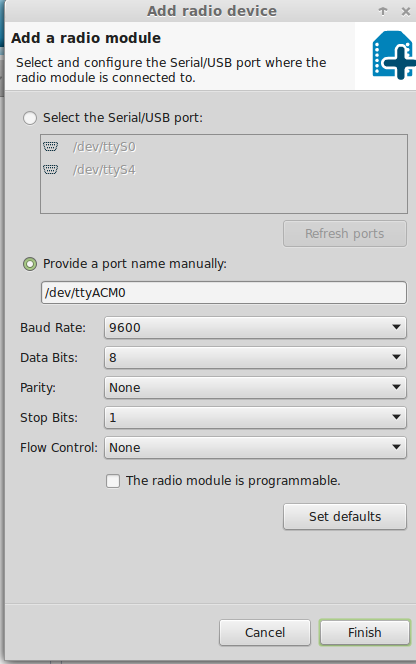
Next you connect the Arduino to the PC and open the XCTU software.

When you open the software. At the top left corner select the “Add a radio module specifying the port settings”.

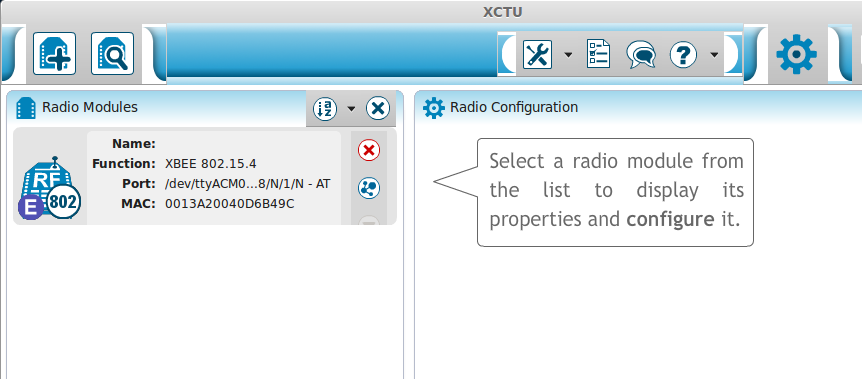


From the window that will pop select “Provide a port name manually” and put the serial port that the Arduino is connected. For me is /ttyACM0 because i was using Linux. To find in Linux the serial port that the Arduino is connected type in terminal “dmesg | grep tty” (without the “ ”). To find the serial port in Windows go to Device Manager and see what COM port your Arduino is connected. For Mac OS it should be something like this “/dev/cu.usbmodemfd131 or 141 “ but look at here for more information on how to find the right name of the USB : <https://support.apple.com/en-us/HT202875> . You can also try the auto detection option of your Xbee .

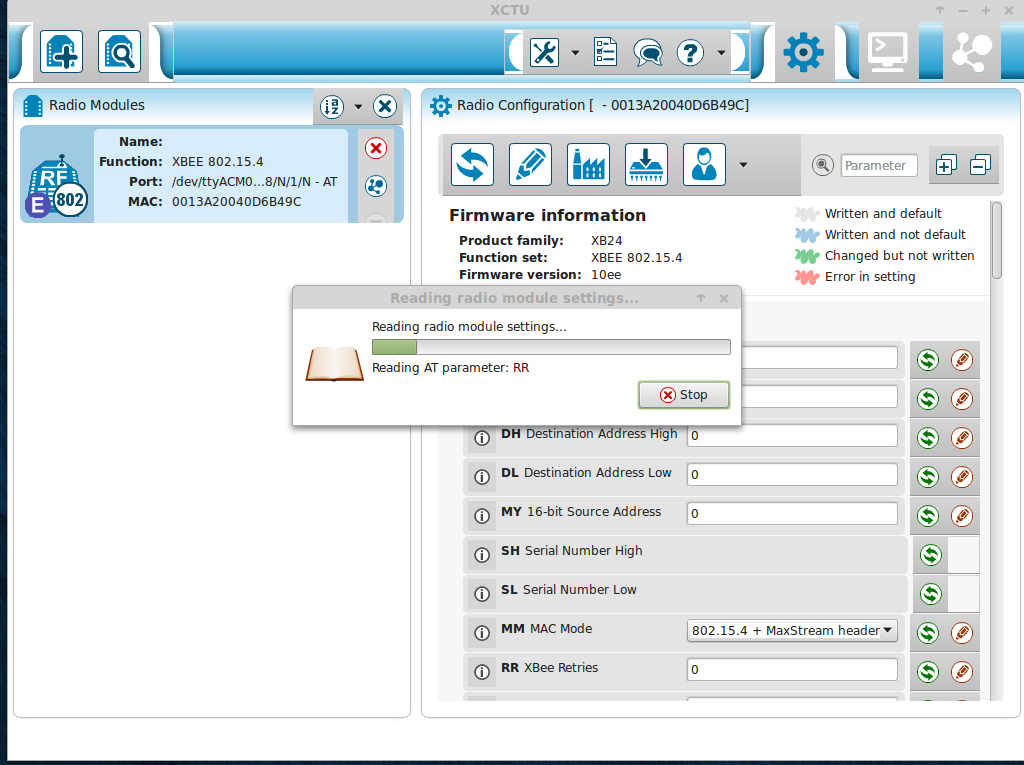
Fill the gaps as appears below. The serial port that you have, Baud Rate at 9600, Data Bits at 8, Parity at None, Stop Bits 1, Flow Control at None. And then click finish.



The next window should pop up:



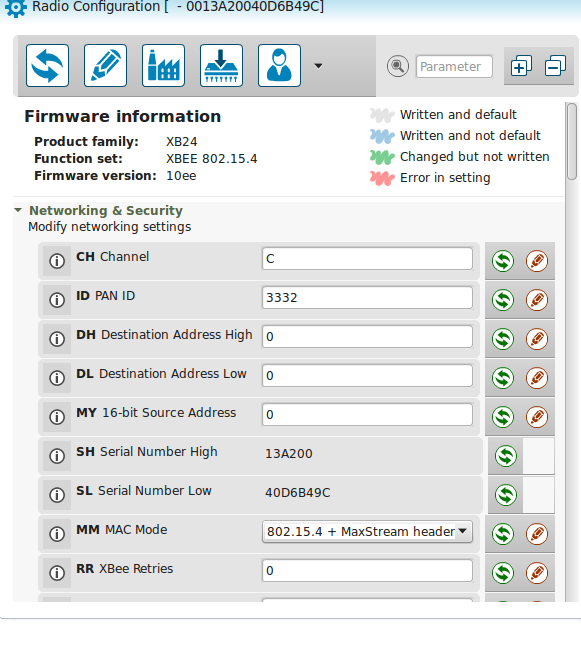
Select the Xbee that it found. The default setup should start on it’s own.



This set up we are gonna need for this project. Because the connection we want is peer to peer, the default setup does exactly that. The only thing you will to pay attention is that the channel CH and the PAN ID ID for both setups must be the same.

For me the Channel CH is C and the PAN ID is 3332.

After that the setup is done, you close the window and disconnect the Xbee.



For the second Xbee you change it and connect it to the Arduino. You follow the same steps to setup it. After that it’s ready to start the wireless communication for your project.

**3.Assembling Controller**

Things we are going to need here:

**Software**

Arduino IDE (<https://www.arduino.cc/en/main/software>). Select the one that matches your OS. This software will be used to program the arduino.

Code for uploading to the Arduino Uno :

All the codes that will be used are in this [link](https://drive.google.com/file/d/0B_OLMcuJEJkcdWVyYXZWQmxiaVU/view?usp=sharing)

(Remember that you can't upload any code to the Arduino when the Tx , Rx pin are connected to the Xbee)

Name of the file : “TFS\_input.ino”

*const int Button1 = 9;*

*const int Button2 = 8;*

*int State2\_before;*

*int State1\_before;*

*void setup(){*

*pinMode(Button1, INPUT);*

*pinMode(Button2, INPUT);*

*Serial.begin(9600);*

*}*

*void loop(){*

*int State1 = digitalRead(Button1);*

*int State2 = digitalRead(Button2);*

*if (State2==HIGH && State2\_before==LOW){*

*Serial.println("a");*

*delay(200);*

*}*

*if (State1==HIGH && State1\_before==LOW){*

*Serial.println("a");*

*delay(200);*

*}*

*State2\_before=State2;*

*State1\_before=State1;*

*}*

**Hardware**

1x Arduino Uno

1x Xbee S1

1x Xbee breakout board

1x Breadboard

1x USB female connector type A

1x USB male A to B (for programming the arduino)

1X Button and 1X 10kΩ resistor (depending on the number of inputs you want. Every input takes one button and one resistor)

Some Jumper wires

Some Pins (male and female)

5x DIP switches , including 2 additional switches on each one. Each DIP switch has 4 pins .Pins 1 and 2 are closer to the OFF counting from the left to the right state and Pins 3 and 4 are in the ON state counting from the left to the right

Assembling the Circuit

**Xbee :**

Solder the Xbee on the breakout board and solder male pins on the outer holes so that the Xbee can connect on the breadboard easily.

Pin 1 = DIP switch 5 to the 1 pin

Pin 2 = DIP switch 4 to the 2 pin + DIP switch 3 to the 2 pin

Pin 3 = DIP switch 4 to the 1 pin + DIP switch 3 to the 1 pin

Pin 10 = DIP switch 5 to the 2 pin

**Arduino :**

3.3V = DIP switch 5 to the 3 pin

5V = Button pin 1

GND = DIP switch 5 to the 4 pin + 1KΩ resistor pin 2 + Pin 5 of the Arduino

Pin 0 = DIP switch 3 to the 4 pin + DIP switch 4 to the 3 pin

Pin 1 = DIP switch 3 to the 3 pin + DIP switch 4 to the 4 pin

Pin 5 = 1KΩ resistor pin 2 + GND

**USB female connector :**

VCC = -

D- = DIP switch 1 to the 4 pin

D+ = DIP switch 2 to the 3 pin

GND = DIP switch 1 to the 3 pin

**DIP switch 1 :**

Pin 1 = GND

Pin 2 = DIP switch 3 www.hobbytronics.co.uk/raspberryto the 1 pin + DIP switch 4 to the 1 pin

Pin 3 = GND of the USB female connector

Pin 4 = D- of the USB female connector

**DIP switch 2 :**

Pin 1 = DIP switch 3 to the 2 pin + DIP switch 4 to the 2 pin

Pin 2 = -

Pin 3 = D+ of the USB female connector

Pin 4 = -

**DIP switch 3 :**

Pin 1 = DIP switch 1 to the 2 pin + DIP switch 4 to the 1 pin

Pin 2 = DIP switch 2 to the 1 pin + DIP switch 4 to the 2 pin

Pin 3 = Pin 1 of the Arduino + DIP switch 4 to the 4 pin

Pin 4 = Pin 0 of the Arduino + DIP switch 4 to the 3 pin

**DIP switch 4 :**

Pin 1 = DIP switch 3 to the 1 pin + Pin 3 of Xbee

Pin 2 = DIP switch 3 to the 2 pin + Pin 2 of Xbee

Pin 3 = DIP switch 3 to the 4 pin

Pin 4 = DIP switch 3 to the 3 pin

**DIP switch 5 :**

Pin 1 = Pin 1 of the Xbee

Pin 2 = Pin 10 of the Xbee

Pin 3 = 3.3V of the Arduino

Pin 4 = GND of the Arduino

**Button :**

Pin 1 = 3.3V Arduino

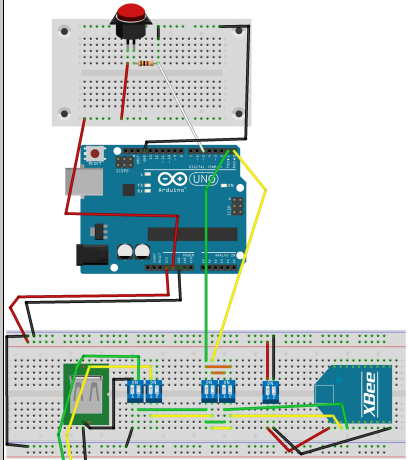
Pin 2 = 330 Ω resistor pin 1

**Resistor :**

Pin 1 = Pin 2 of the Button

Pin 2 = Pin 5 of the Arduino + GND of the Arduino

**It has to look like this :**



Uploading Code

(Remember that you can't upload any code to the Arduino when the Tx , Rx pin are connected to the Xbee)

When everything is set you can upload the code to the Arduino via the Arduino IDE program :

Connect the Arduino board to the computer with the USB cable A to B type and wait until the LED s on the board stabilizes .

Compile the the code by clicking the check button on the top left next to the right-pointing-arrow button.

Select the the USB port of the computer which the Arduino is connected from Tools -> Serial Port. For Windows users : the name of the USB port indicates the number of the specific USB port that the Adruino board is connected and in most cases there Arduino has been recognized and his name is written in parentheses next to the name and number of the USB port . For Linux users : the name of the USB port in which the Arduino board is connected looks like this : “ /dev/ttyAMC0 “ or “/dev/ttyAMC1” .For MAC see here how to connect the Arduino : <https://www.arduino.cc/en/Guide/MacOSX>

Select Tools -> Programmer -> AVRISP mkll

Finally make sure the your model of the Arduino board is selected (Uno) from Tools -> Board -> Arduino Uno

Switch OFF both switches of DIP switch 3 .

Upload the code clicking on the right-pointing-arrow button in the top left.

Switch ON both switches of DIP switch 3 .

**4.Assembling Control Unit**

Things we are going to need here:

**Software**

1. At first we need to we need to put an image in the Raspberry Pi. We used the Raspbian, you can find the image and download it [here](https://www.raspberrypi.org/downloads/raspbian/) : (<https://www.raspberrypi.org/downloads/raspbian/>).

There is a good installation guide of how you are going to put in the Micro SD card. The [link](https://www.raspberrypi.org/documentation/installation/installing-images/README.md) for the guide (<https://www.raspberrypi.org/documentation/installation/installing-images/README.md>)

2. We need to set the Pi to boot to Desktop Environment. We can do this setting either from the forst time we boot it or if skipped it open up a terminal and write:

$ sudo raspi-config

And select to boot directly to Desktop.

3. After that you need to change the keyboard layout of the Raspberry because the “GB” layout is by deafault. Open up a Terminal and write:

$ sudo dpkg-reconfigure keyboard-layout

Select the US layout. And reboot the Pi to apply the changes.

4. Next step is to enable the Serial Port of the Raspberry Pi, this way we can connect the Xbee and communicate with it. There is a good Guide of how we are going to do that (<http://www.hobbytronics.co.uk/raspberry-pi-serial-port>)

First,open up a Terminal and write:

$ cd /etc/

$ sudo nano inittab

This file has the command to enable the login prompt and this needs to be disabled. Edit the file and move to the end of the file. You will see a line similar to

T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100

Disable it by adding a # character to the beginning. Save the file.

#T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100

The next file you need to modify is (open a new Terminal):

$ cd /boot/

$ sudo nano cmdline.txt

The contents of the file look like this dwc\_otg.lpm\_enable=0 console=ttyAMA0,115200 kgdboc=ttyAMA0,115200 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait

Remove all references to ttyAMA0 (which is the name of the serial port). The file will now look like this dwc\_otg.lpm\_enable=0 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait

Now it’s time to make the python code run:

Go to this [link](https://drive.google.com/open?id=0B_OLMcuJEJkcaURkdk5fTnJtU2c) and download the python code (<https://drive.google.com/open?id=0B_OLMcuJEJkcaURkdk5fTnJtU2c>)

The code is:

*import RPi.GPIO as GPIO*

*import time*

*import serial*

*Button=21*

*delaytime = 0.03 # psaxnoume gia to grhgorotero on kai off*

*ser = serial.Serial('/dev/ttyAMA0', 9600)*

*GPIO.setmode(GPIO.BCM)*

*GPIO.setwarnings(False)*

*GPIO.setup(Button, GPIO.OUT, initial=0)*

*while True:*

*incoming = ser.readline().strip()*

*if incoming == 'a':*

*GPIO.output(Button, 1)*

*tim.sleep(delaytime)*

*GPIO.output(Button, 0)*

*print('You pressed : %s' % incoming)*

*else:*

*GPIO.output(Button, 0)*

*print('You pressed Nothing')*

*GPIO.cleanup()*

To run the code open up a Terminal (let’s say you have to your /home/pi folder) and write:

$ cd /home/pi/

$ sudo python ControlUnitPython.py

The code should be running.

6. The last step from the software is to make the python script run on startup at the background.

The guide of how you make this happen is [this](http://www.raspberrypi-spy.co.uk/2014/05/how-to-autostart-apps-in-rasbian-lxde-desktop/). (<http://www.raspberrypi-spy.co.uk/2014/05/how-to-autostart-apps-in-rasbian-lxde-desktop/>)

Save the python script at the pi folder.

Open up a Terminal and write:

$ cd /etc/xdg/lxsession/LXDE-pi

$ sudo nano autostart

At the end add a line by writing:

@sudo python /home/pi/ControlUnitPython.py

Save it and reboot the Pi. The script should be running at the background.

**Hardware**

1x Raspberry Pi (model doesn’t really matter but it would be better to get Raspberry 2)

1x USB micro for power

1x micro SD card at least 4GB

1x micro SD card at least 4GB for Pi 2 -or- 1x SD card at least 4GB for Pi 1

1x SD card Adapter for accesing into the SD card or micro SD card

1x HDMI

1x Xbee S1

1x Xbee breakout board

1x Breadboard

1x USB female connector type A

1x LED

1X 330 Ohms resistor

Some Jumper wires

Some Pins (male and female)

4x DIP switches , including 2 additional switches on each one. Each DIP switch has 4 pins .Pins 1 and 2 are closer to the OFF state counting from the left to the right and Pins 3 and 4 are in the ON state counting from the left to the right.

Assembling

**Xbee :**

Solder the Xbee on the breakout board and solder male pins on the outer holes so that the Xbee can connect on the breadboard easily.

Pin 1 = DIP switch 4 to the 1 pin

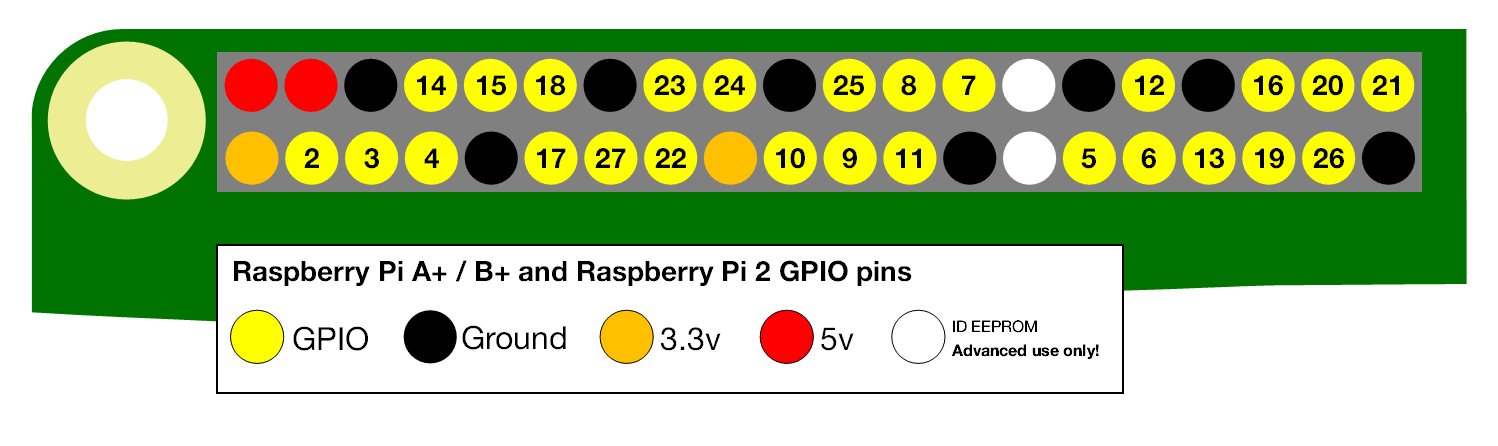
Pin 2 = DIP switch 3 to the 1 pin + DIP switch 2 to the 1 pin

Pin 3 = DIP switch 3 to the 2 pin + DIP switch 1 to the 2 pin

Pin 10 = DIP switch 4 to the 2 pin

**Raspberry Pi :**

The following image shows the names of the Raspberry's GPIO pin. The image has the names of the pins which the Raspberry understands originally . This means that if you want to refer to them while programming the Raspberry you will use this coding for the GPIO pins.



3.3V = DIP switch 4 to the 3 pin

Ground = DIP switch 4 to the 4 pin + Cathode of the LED +

DIP switch 1 to the 1 pin

Pin 14 (TXD) = DIP switch 3 to the 4 pin

Pin 15 (RXD) = DIP switch 3 to the 3 pin

**USB female connector :**

VCC = -

D- = DIP switch 1 to the 4 pin

D+ = DIP switch 2 to the 3 pin

GND = DIP switch 1 to the 3 pin

**DIP switch 1 :**

Pin 1 = Ground of Raspberry

Pin 2 = DIP switch 3 to the 2 pin + Pin 3 of the Xbee

Pin 3 = GND of the USB female connector

Pin 4 = D- of the USB female connector

**DIP switch 2 :**

Pin 1 = DIP switch 3 to the 1 pin + Pin 2 of the Xbee

Pin 2 = -

Pin 3 = D+ of the USB female connector

Pin 4 = -

**DIP switch 3 :**

Pin 1 = DIP switch 2 to the 1 pin + Pin 2 of the Xbee

Pin 2 = DIP switch 1 to the 2 pin + Pin 3 of the Xbee

Pin 3 = Pin 15 (RXD)

Pin 4 = Pin 14 (TXD)

**DIP switch 4 :**

Pin 1 = Pin 1 of the Xbee

Pin 2 = Pin 10 of the Xbee

Pin 3 = 3.3V of the Raspberry Pi

Pin 4 = Ground of the Raspberry Pi

**Resistor :**

Pin 1 = Pin 6 of the Xbee

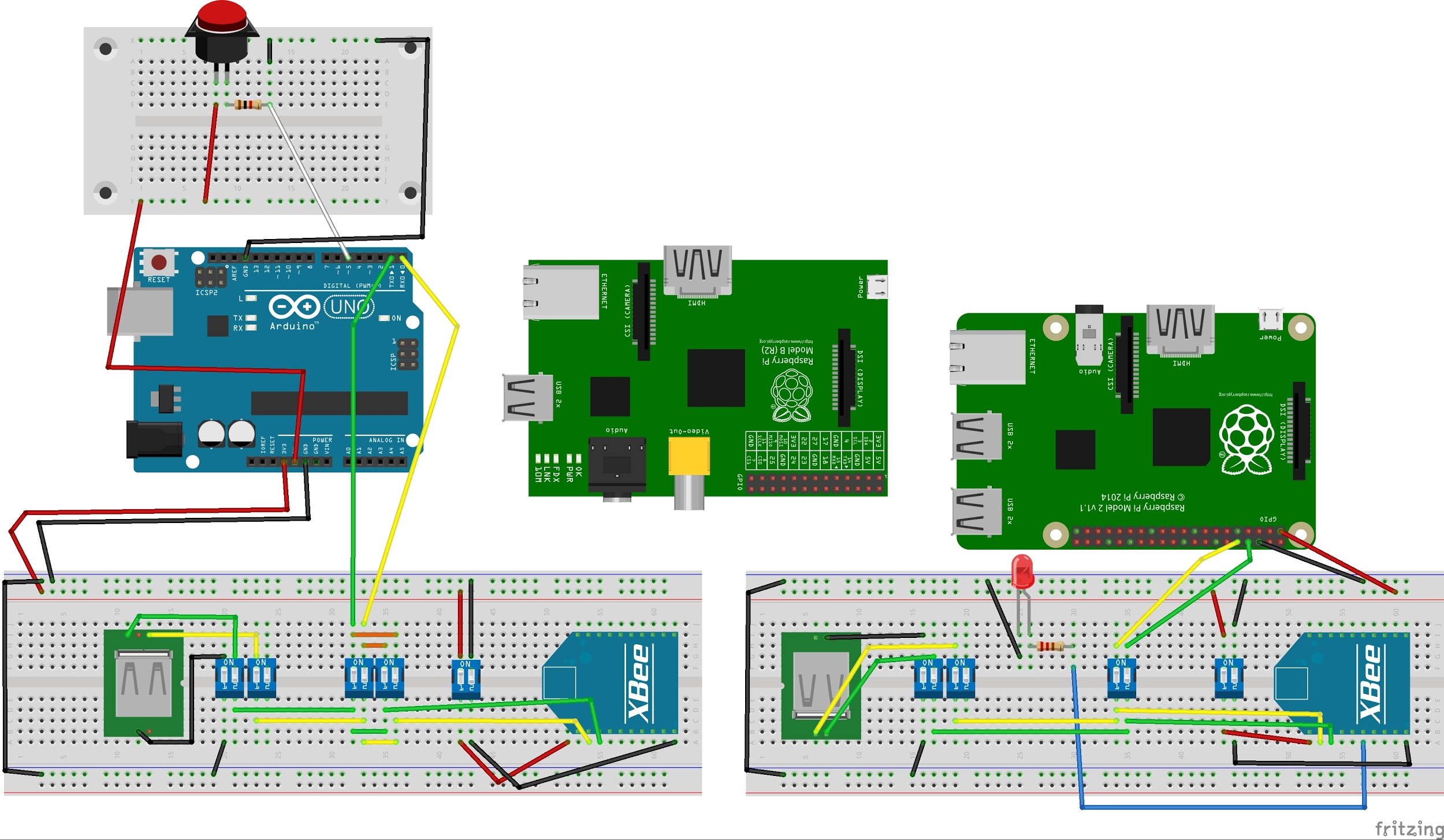
Pin 2 = Anode of the LED

**LED :**

Anode = Pin 1 of the resistor

Cathode = Ground of the Raspberry Pi

**It has to look like this :**



(The Raspberry Pi 1 in the middle is there only to indicate the TXD and RXD on the pins 14 and 15 which are in the same position on the Pi 2)

Uploading Code

APO TA SOFTWARIKA ERGALIA KANOUME TA E3HS GIA NA LEITOURGISEI

Transfer the the codes and the ThinkFreedom program to the Raspberry Pi from the computer with the help of a USB stick.

8a pas ekei kai 8a katebaseis auta to arxeio. 8a pareis auto to arxeio kai 8a to peraseis se ena usb gia na to baleis sto raspberry.

apo auto to link 8a pas kai 8a akolou8hseis tis odhgies “ setting up the development enviroment” wste to raspberry na einai etoimo gia thn egkatastash tou programmatos :

<https://github.com/scify/JThinkFreedom>

**5.How it works**

After the assembling is complete the system is ready to work . By pussing the button the arduino converts this digital signal into a letter (character or char) . The entirelychar is sent from the serial port of the Arduino , with the help of the Xbee or USB cable male to male , to the serial port of the Raspberry . The Raspberry reads this exact letter .Every time it reads a specific letter decides to give a specific command to the ThinkFreedom program to react . Adding different buttons to the system , you have to assign them to different letters and adding different cases when one of them is pressed. The rule of the switches is to alternate from wireless communication to wired and to set up the Xbees without removing them from the assembly . The switches tries to imitate approximately an integrated circuit that does the same work automatically .

Wireless use :

**Turn ON (up)** : DIP switches 3 and 5 from the controller side

DIP switches 3 and 4 from the control unit side

**Turn OFF (down)** : DIP switches 1 , 2 and 4 from the controller side

DIP switches 1 and 2 from the control unit side

Wired use :

**Turn ON (up)** : DIP switches 1 , 2 and 3 from the controller side

DIP switches 1 , 2 and 3 from the control unit side

**Turn OFF (down)** : DIP switches 4 and 5 from the controller side

DIP switches 4

**Setting up the Xbee from the controller side :**

This way is an additional way ( compared to the first one that has been referred before with title “*Setting up the Xbee*”) of setting up the Xbee from the controller side when you have already assembly all the parts and you do not want to involve with the circuits any more .

**Turn OFF (down)** : Everything

Upload the blank code to the Arduino following the previous instructions at the “*Setting up the Xbee*” chapter.

**Turn ON (up)** : DIP switches 5 and 4 from the controller side

**Turn OFF (down)** : DIP switches 1 , 2 and 3 .

Run the XCTU program from Digi as it has been already described before in the “Setting up the Xbee” chapter .

(After these steps do not forget to upload the main code to the Arduino (again) )

**Setting up the Xbee from the control unit side :**

This way is an additional way ( compared to the first one that has been referred before with title “Setting up Xbees”) of setting up the Xbee from the control unit side when you have already assembly all the parts and you do not want to involve with the circuits any more .

From the controller side :

**Turn OFF (down)** : Everything

From the control unit side :

**Turn OFF (down)** : Everything

Upload the blank code to the Arduino following the previous instructions at the “*Setting up the Xbee*” chapter.

From the controller side :

**Turn ON (up)** : DIP switches 1 , 2 and 3 from the controller side

**Turn OFF (down)** : DIP switches 4 and 5 from the controller side

From the control unit side :

**Turn ON (up)** : DIP switches 1 , 2 and 4 from the control unit side

**Turn OFF (down)** : DIP switches 3 from the control unit side

Run the XCTU program from Digi as it has been already described before in the “Setting up the Xbee “ section .

(After these steps do not forget to upload the main code to the Arduino (again) )

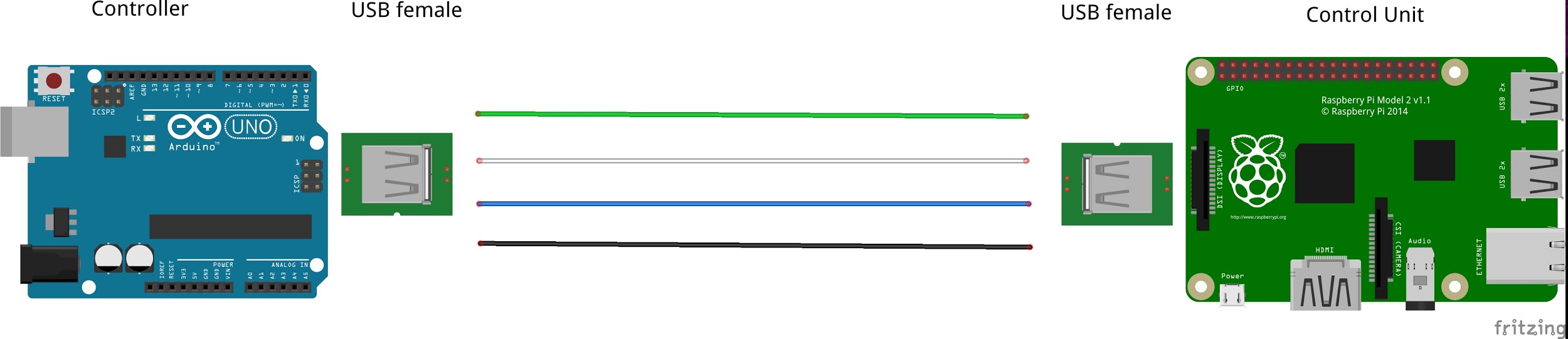
**6. Further Upgrades**

**1.**

Logic Circuit that Decides to Switch Between Wireless and Wire Mode :

The DIP switches can be replaced by a specific logic circuit that will do the same job. Considering that the setup procedure will take place independently , before the assembly of the final circuits , the logic circuit could work like this :

The four wires of the USB 2.0 will be used in a different way than the prototype.



Green : Tx from the controller (Arduino UNO pin 1)

White : 3.3 V from the control unit side to the controller’s logic circuit as an

input signal that indicates the cable connection.

Blue : 3.3 V from the controller to the control unit’s logic circuit as an input

signal that indicates the cable connection.

Black : Common Ground

(Be careful the Xbee operates only on 3.3 V !)

(Also , be aware that the TXD and RXD in on the one end of the USB are reversing inside the cable to the other end. So the Blue and White cables inside the USB are not as presented at the above image . You have to make another costume reversing of your own ).

From the Controller side the Inputs and Outputs of the logic circuit :

IN : White cable of the usb

**1** : when USB is connected.

**0** : when USB is disconnected

OUT : Power Supply to the Xbee only on 3.3V   
 **1** : when USB is disconnected.   
 **0** : when USB is connected

From the Control Unit side the Inputs and Outputs of the logic circuit :

IN : Blue cable of the USB

**1** : when USB is connected.

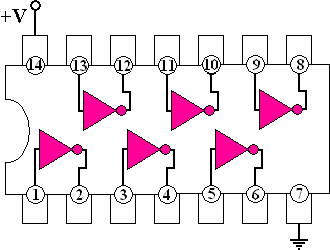
**0** : when USB is disconnected

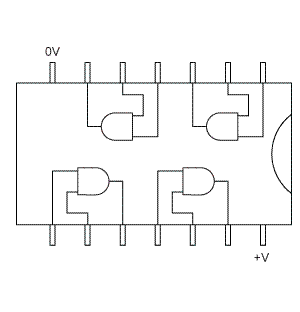
OUT : Power Supply to the Xbee only on 3.3V

**1** : when USB is disconnected.

**0** : when USB is connected

Both logic circuits can be build with one NOT logic gate and one AND logic gate :





(The images above show one integrated circuit with 6 NOT logic gates and one integrated circuit with 4 NAND logic gates)

*To build the circuits on the controller side :*

Make the necessary adjustments to the USB female connector so that you know which cable does what.

Connect the White wire to the Input of the NOT gate .The White wire of the USB cable has 3.3V coming from the Raspberry Pi.

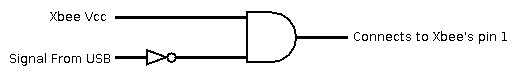
Connect the Output of the NOT gate to the one of the inputs of the AND gate.

Connect the 3.3V of the Arduino to the other input of the same AND gate.

Connect the Output of the AND gate to the 3.3V pin 1 of the Xbee.

Finally , Connect the power supplies of 3.3V and the grounds to the integrated circuits in cause you have got two separate like shown in the image above.

(Remember that you can't upload any code to the Arduino when the Tx , Rx pin are connected to the Xbee)



*To build the circuits on the control unit side :*

The procedure is similar to the previous.

Done.

Note 1:

To take this upgrade to another level you can use the White Wire coming from the Raspberry Pi at 3.3V to recharge the battery of the controller side.You can do that by spliting the White Wire into two wires when youa re looking the controller side . The one wire can go to the logic circuit as it has been described before . The other one can go to a new circuit that recharges the battery.

Note 2:

All the above have not been tested so in the real world could occur some practical difficulties . In this case you have to reissue the design.

**3.**

3D Tracking Interface for Input :

<http://makezine.com/projects/a-touchless-3d-tracking-interface/>

**4.**

Camera module :

A camera module can be used to detect facial movements . In cases where the abilities of the patient is really limited this is ideal .

Changes in the hardware and software must take place .

It is necessary to easily add the camera without further more adjustments in hardware or software . So the hardware has to change in a way where there will be a female connector for the module “waiting” a plug-in to happen and then automatically adapt to it. The software has to change and have the sensor class of the camera and execute it when it will plug in .

This is the camera module :

<https://www.raspberrypi.org/help/camera-module-setup/>

**5.**

More Buttons :

In same cases the joystick could be a perfect solution for navigation.

The hardware and software changes are very easy to do.

You have to add more button pins , just like the existing one , on the controller side and then connect them to a USB female connector waiting for a plug in . You have to relate the new buttons with new if cases in the software of the controller side as well as in the control unit side . The ThinkFreedom program should have the flexibility for manual navigation when needed while still having the time lapse navigation.l