

# Muzzley Intel Galileo Demos

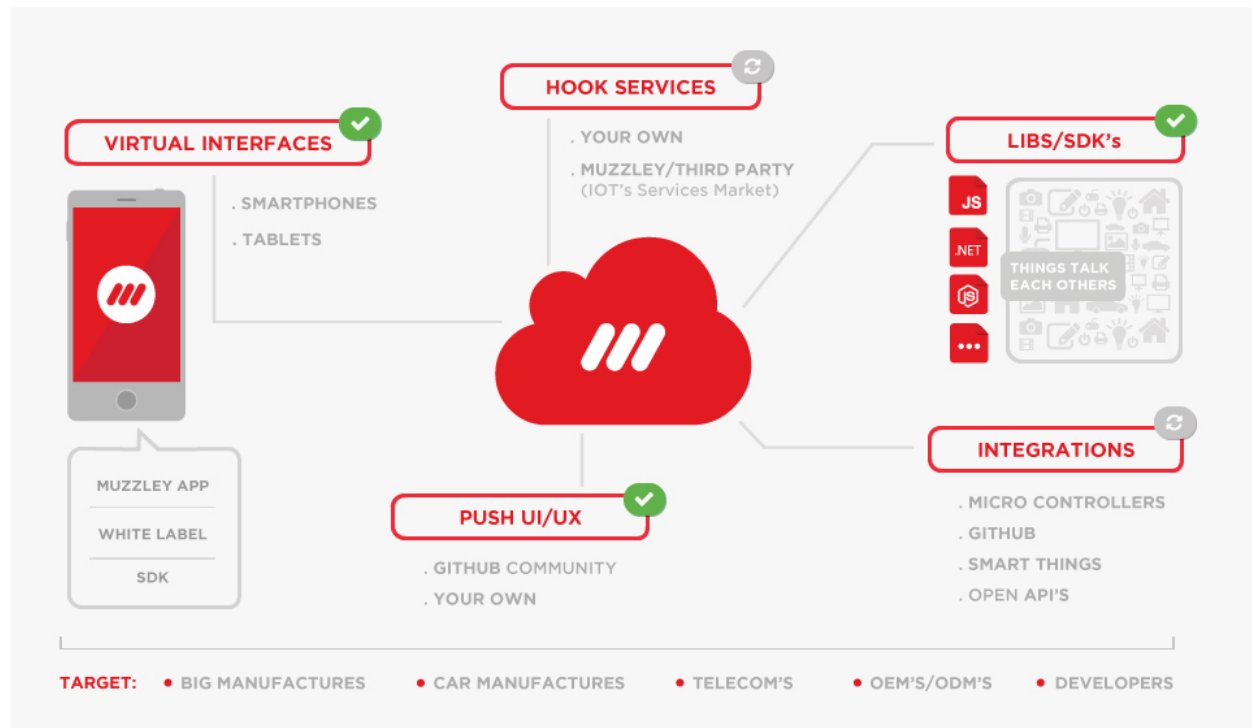
## Intro

The main goal of this demo is to explain how you can control your Intel Galileo board and its applications with a mobile device using Muzzley. This walkthrough explains in detail how you can control a digital rgb LED strip and the Galileo board pins using a mobile device.

This demos were built with Node.js and require the Galileo to run with a Linux image in the micro SD card.

## Why Muzzley?

Muzzley is a technology that allows mobile control integration, making it easier and faster for developers, also greatly increasing the user experience.



In a very brief description, it allows you to develop your smartphone/tablet customized interface and enables a message exchange system between your application and your smartphone, making anything you need available on your device. Another good thing is that it is free for use.

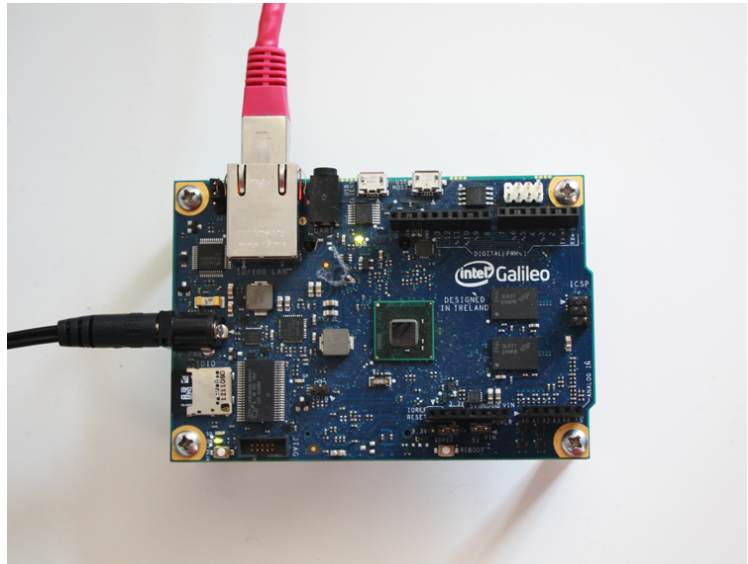
The views are loaded into your mobile devices by pairing with your application Token / QR code.

## Hardware requirements

The following equipment is required in order to run the demo

- Intel Galileo Board

Galileo is a micro controller board based on the Intel® Quark SoC X1000 Application Processor, a 32-bit Intel Pentium-class system on a chip. It's the first board based on Intel® architecture designed to be hardware and software pin-compatible with Arduino shields designed for the Uno R3. The pins are all in the same locations as on the Arduino Uno R3. This is also known as the Arduino 1.0 pinout. Galileo is designed to support shields that operate at either 3.3V or 5V.



([http://www.intel.com/newsroom/kits/quark/galileo/pdfs/Intel\\_Galileo\\_Datasheet.pdf](http://www.intel.com/newsroom/kits/quark/galileo/pdfs/Intel_Galileo_Datasheet.pdf))

- Micro SD Card

A micro SD card is required in order to use Linux as the operating system, it is suggested at least one with 512MB. If your computer doesn't have a micro SD drive you'll probably need an adapter.



- RGB Digital Led Strip

In this demo it was used a Digital RGB LED Waterproof Strip 98 LED (<http://www.adafruit.com/products/306>), but it is possible to use any kind of digital LED strip as long as they use the LPD8806 protocol.



- Power Supply

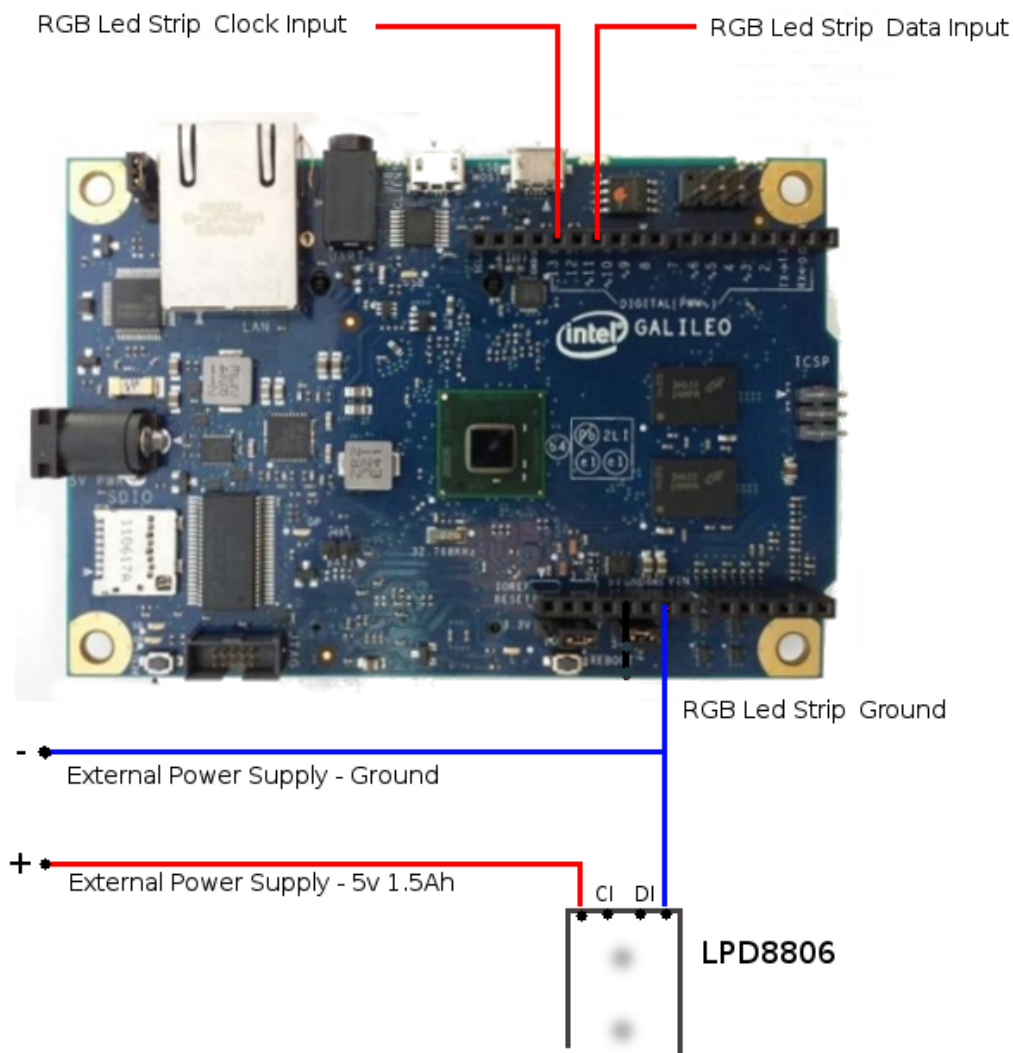
It will be necessary at least one PSU to power up the Galileo board. We also had to use another power supply (5V DC) because the strip was long and required more amperes in order to be able to flash the white colour correctly. Shorter led strips can connect straight to the on-board 5V pin.



## Hardware Setup

In the image below, a diagram of how to connect the digital LED strip to the Galileo using an external power supplier is shown:

- Galileo's Clock output is pin 13 and should connect to the LED strip Clock Input (CI)
- Galileo's Data output is pin 11 and should connect to the LED strip Data Input (CI)
- The maximum recommended voltage for the digital LED strip is 5v
- If using an external power supply for the LED strip, ground must be shared between the PSU and the Galileo ground



## Software Requirements

In order to run the demos you'll just need to download or create a Linux image in the SD card containing Node.js, npm and its dependencies. This image includes the wifi drivers.

### Setting up the SD card with the provided image

1. Update your board firmware if not updated already
2. Insert your SD card in the drive, use an adapter if necessary
3. Format the card
4. Download the image file from -----
5. Unzip the downloaded file into the formatted SD card  
\*If using Linux check the device name with the command "fdisk -l" then execute "sudo gzip -dc /home/user\_example/muzzley.galileo\_20140217.gz | dd bs=4M of=/dev/sdb" where /dev/sdb is the device given by "fdisk -l"
6. Unmount the SD card
7. Insert the SD card in the Galileo and power it
8. The image contains a watchdog and it is configured to start by itself

### Create your Muzzley account

1. Create your Muzzley account at [www.muzzley.com](http://www.muzzley.com) if you don't have one already
2. Create your app and name it
3. Create a widget
4. Open the widget editor
5. Copy widget/widget.js contents to JavaScript field in the editor
6. Copy widget/widget.html contents to HTML field in the editor
7. Copy widget/widget.css contents to CSS field in the editor
8. By default Muzzley activities are dynamic, if you wish to always use the same activity you must generate a static one in the static activities tab (Muzzley site > Apps > My App).
9. Save your widget

### Setting up the application

1. Find your Galileo IP address
2. Open a terminal and type `ssh root@MY_GALILEO_IP`
3. Edit the config file by typing `'vi /home/projects/intel-galileo-demos/config.js'`
4. In the editor, locate the following object:  

```
ledStripe: {  
  ledsNum: 97,  
  spiDevice: '/dev/spidev1.0'
```

- ```
}
5. Adjust your LEDs number by changing the var ledsNum
6. Locate the following object:
muzzley: {
  token: "123456789.."
}
7. Update the token var with you app appToken (given at Muzzley website), if you generated
a static activity, add the following key to the Muzzley object:
muzzley: {
  token: "123456789..",
  activityId: "My_static_activity"
}
Where the activityId is the the static one created in the website.

8. Locate the object:
widgets:{
  wmcInterface: "abcdef-ghij-4441-aaaa-bbbbbbb3a"
}
9. Update the var wmcInterface to your widget id (check widget details in Muzzley website)
```

After changing the application config file, it is required to restart the app. In order to do so:

- Use the command “/etc/init.d/muzzley-galileo stop” to stop the application
- Use the command “/etc/init.d/muzzley-galileo start” to start the application again with the new configuration loaded

## Running the demos

As soon as you watch the leds start playing an animation, it means the muzzley application has started and that you are ready to pair your device with it.

Use your muzzley mobile application and pair with the program. If you are using a static activity, you can insert the activity code straight away or scan the qrcode at [http://www.muzzley.com/qrcode/my\\_static\\_activity](http://www.muzzley.com/qrcode/my_static_activity), where my\_static\_activity is your static activity. If you are not using a static activity you need to find the generated activityId in the logs, in order to do so, execute the command “tail -f /var/log/muzzley/intel-galileo-demos.log -n 30”, and locate the following line “[info] Connected. activityId: My\_activity\_id”. Insert in the Muzzley mobile app the activity id.

After the pair process you should have a visual interface in your mobile display. Multiple simultaneous participants are allowed, but only one animation will be played at a time.

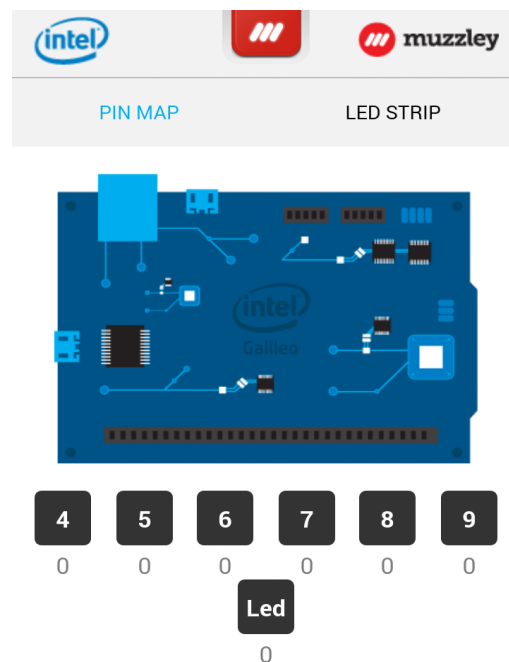


## The Galileo pin debugger

This demo allows the user to read and write to the galileo board pins. Allowed values are 0 and 1 and the allowed directions are In and Out. Beware that you can only change a value if the direction is set to Out.

By default the galileo pins 4, 5, 6, 7, 8, 9 and the on-board LED were made available (GPIOs 28, 17, 24, 27, 26, 19, 3).

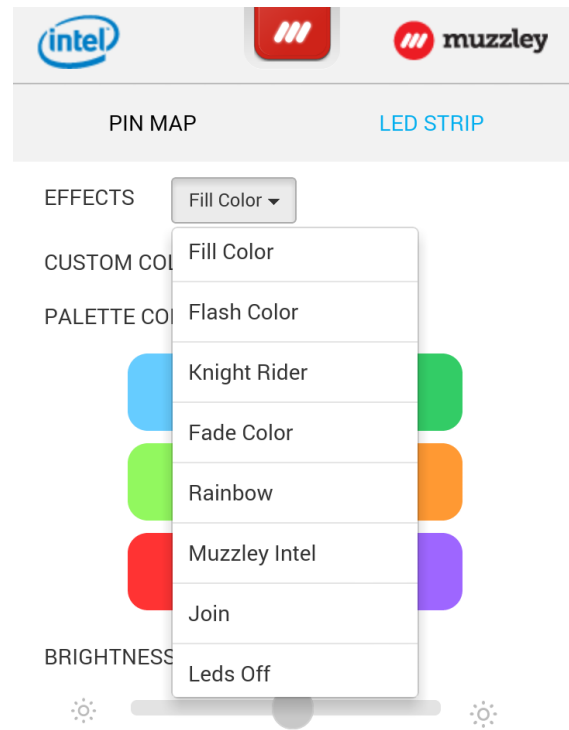
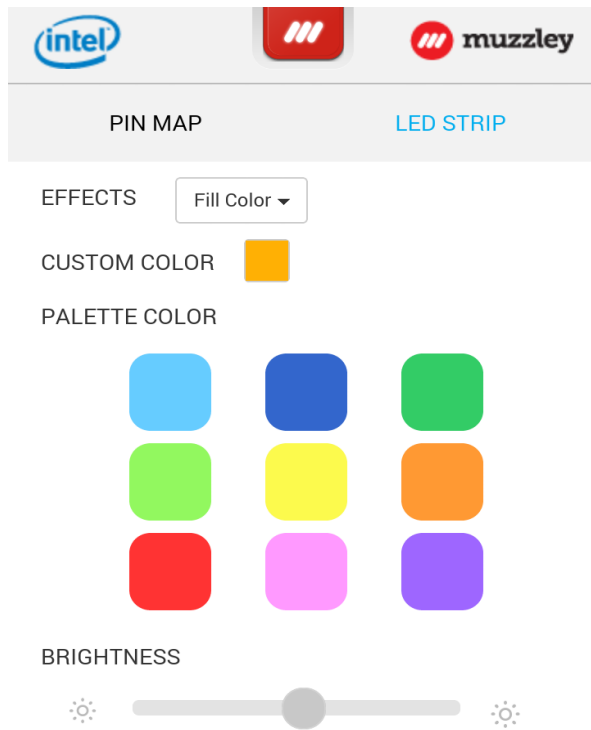
Clicking on a pin will show you a popup which will allow you to do such operations. You can try for instance connecting an LED to a specific pin and use the mobile device to turn it on and off setting the direction to out and switching between values 0 and 1.



## The Galileo LED strip controller

This one allows the user to control a LED strip. You can fill it with colors or play some colorful animations. Picking the color, selecting an effect or adjusting brightness will trigger the animation with the selected parameters. The color palette is available on clicking the custom color button,

you can try slowly dragging your finger in the color palette and watch the LED strip changing colors in real time.



## Paths

- The project directory is /home/projects/intel-galileo-demos
- The application log file is accessible using the command “tail -f /var/log/muzzley/intel-galileo-demos.log”



## Known Issues

- We are using Node.js to manipulate the LEDs. In order to make the code nonblocking, we had to make the animations asynchronous. Sometimes the animations may not run at the same speed.
- The animation queue is very small, if you have multiple participants connected and queueing a long number of animations in a short time period, the first animations in the queue will be discarded.
- In order to read the GPIO's information, we are polling them each 5s, having multiple participants at same time, may have a bit of delay updating the visual information in the other participants' views (It is possible to set the poll time shorter).
- The first led of the strip is always off.