Multi-Agent Decision Making Over Wireless Networks

Project Report

Insight Centre for Data Analytics

University College Cork

Martin Bullman

Summer Intern

martin.bullman@insight-centre.org

From 11th June – 2nd September 2014

Supervisor: Ken Brown (k.brown@cs.ucc.ie)

Mentor: Mohamed Wahbi (mohamed.wahbi@insight-centre.org)

Table of Contents

This is a report on my summer Internship at Insight Centre for Data Analytics at University College Cork. During my twelve-week Internship, I was tasked with completing a “Multi-Agent Decision Making over Wireless Networks” project. While I was an Intern at Insight I was working under the supervision of Kenneth Brown and my mentor was Mohamed Wahbi.

Introduction

Real-time decision-making based on heterogeneous data is becoming increasingly important in a wide range of industry sectors. For many applications in existence today this involves gathering data from widely distributed sensors, sending the data back to a centralised access point, using that data to make appropriate decisions and then transmitting the results back out to remote actuators. Based on the results of the decision process devices could then perform tasks e.g. switch on a light, open a vent, increase airflow in a room etc.

This process of centralised communication is often to slow for real-time decision-making. So what we wanted to try and achieve was to remove this process of centralised communication so we would basically have an application that uses decentralised communication, meaning devices would transmit their data directly to every connected device and make decisions based on that data instead of transmitting the data back to a centralised access point and making the decisions there. Most existing research in the areas of distributed reasoning ignores the issues with wireless sensing, communication and actuation.

The goals I set during my time at Insight were to construct a wireless testbed using multiple Intel Galileo microcontrollers and to have each of the devices communicate wirelessly with other connected devices over an ad-hoc wireless network without the need for a centralized access point. We would then have a wireless platform which I could use to move onto another goal in which I was tasked with constructing a demonstration application on top of this wireless test bed. This involved extending the Galileo microcontrollers with breadboard, sensors and LEDs, then programming a protocol to control this hardware so we could gather data from the surrounding environment, transmit the data to other devices, have each device make a decision based on the data and show the result using LEDs

Before I began working with Intel Galileo I began researching the area of ad-hoc wireless networks to gain knowledge and get a better understanding of how they actually worked. This involved reading ad-networking books, online articles and abstract papers written by other researchers. In my first few weeks, I was lucky enough to attend two presentations which were located on the UCC campus which gave me invaluable insight into the concepts I could use in my project.

What is the Intel Galileo Development Board?

The Intel® Galileo Gen 2 development board, is the first in a family of Arduino*-certified development boards based on Intel® architecture and is specifically designed for makers, students, educators, and DIY electronics enthusiasts.

Intel® Galileo runs a Yocto Linux image as its operating system which has the basic functionality of a normal Linux OS. The Galileo is based on the Intel® Quark™ SoC X1000, a 32-bit Intel® Pentium® processor.

For this project, Intel donated five Intel Galileo development boards for me to prototype with and use as the foundation of the wireless test bed. I was also given five Intel Centrino N-135 Wi-Fi cards and an antenna for each board.

Specifications

400MHz 32-bit Intel® Pentium instruction set architecture (ISA)-compatible processor

- 512 Kbytes of on-die embedded SRAM
- Simple to program: Single thread, single core, constant speed
- An integrated Real Time Clock (RTC)
- 512 Kbytes embedded SRAM that is enabled by the firmware by default.
- 256 Megabytes DRAM, enabled by the firmware by default.
- Optional micro SD card offers up to 32GByte of storage

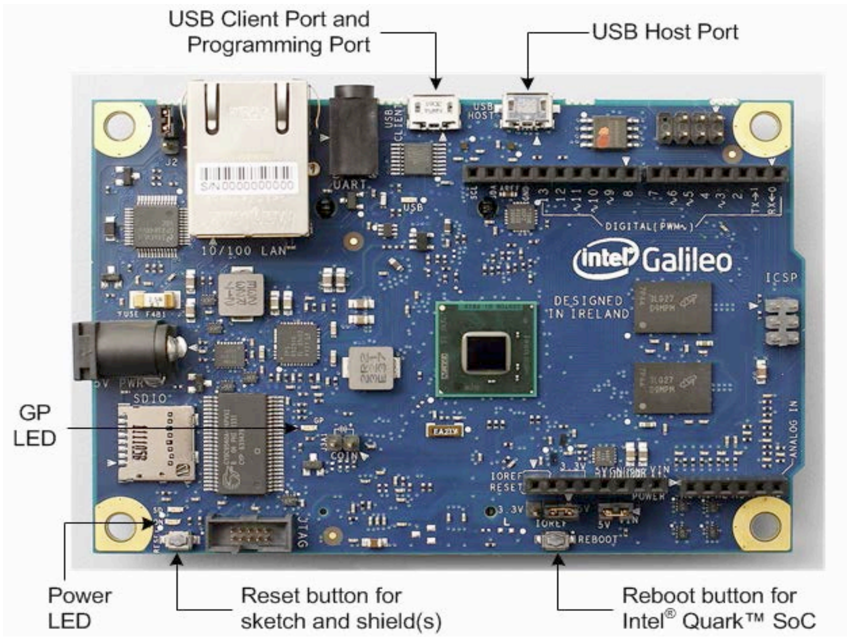

USB Client Port and Programming Port
USB Host Port
GP LED
Power LED
Reset button for sketch and shield(s)
Reboot button for Intel® Quark™ SoC

Figure 1: The Intel Galileo microcontroller development board

Constructing the Intel Galileo Wireless Test Bed

Before I could start constructing the Galileo wireless test bed I had to set up each individual Galileo microcontroller. The steps to do this included:

Installing the Intel Centrino Wi-Fi card

1. To install the Wi-Fi card I first had to attach the expansion plate (shown in fig1) onto the actual Wi-Fi card using the two screws provided.
2. The next step was to clip on the Wi-Fi antenna, as you can see from Fig2 there are two connection slots available one is labelled MAIN the other is labelled AUX. For this project we are not using the Aux connection so we simply connect the antenna to the connection labelled Main (shown in Fig 2)

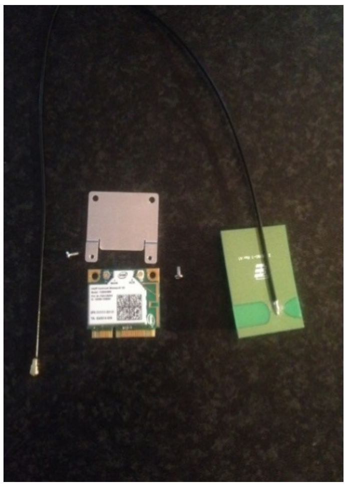


Figure 2 Components used to attach Wi-Fi card to Galileo

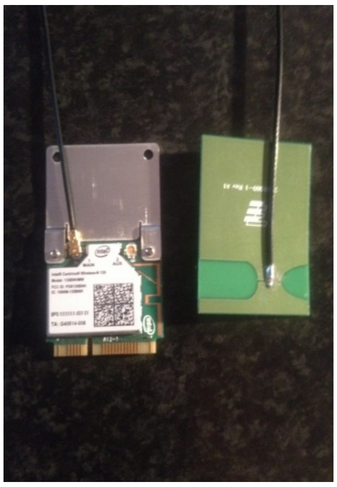


Figure 3 Wi-Fi card with expansion plate and antenna attached

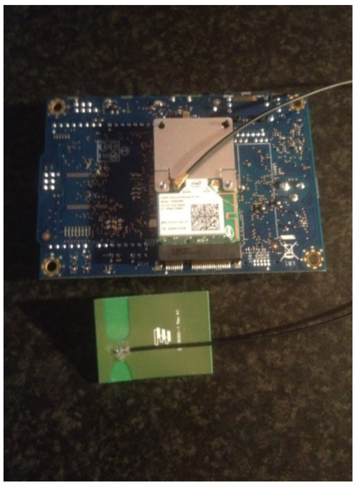


Figure 4 Wi-Fi card attached to the Intel Galileo microcontroller

3. The final step in this process was to install the Wi-Fi card onto the Intel Galileo board itself. To do this slot the Wi-Fi card at a 45 degree angle into the mini PCI slot at the back of the Galileo, then press down on the Wi-Fi card until it clips into place.

After completing the above steps I now have a Galileo board with wireless capabilities. The next thing was to repeat the same process with the other four Galileo boards.

Upgrading the Intel Galileo firmware

To upgrade the firmware I needed to first download the Intel Galileo Arduino IDE from the following link: http://arduino.cc/en/Main/Software. The Arduino IDE enables users to write programs for the Galileo called sketches and directly upload the sketch to the board using the serial port. For this project I only need the Arduino IDE to upgrade the firmware and to run a simple blink sketch to be certain the board is functioning correctly after upgrading the firmware and installing the Wi-Fi card. If you are downloading this IDE make sure you download the correct version as other versions won't be compatible with the Intel Galileo.

Before running the Arduino IDE for the first time I had to connect a mini USB cable between the Galileo and my computer, you may notice there are two mini USB ports on the Galileo so the correct one to use is the port closest to the Ethernet port. This enables the serial port on the Galileo for communicate with my laptop and will enable me to upgrade the firmware easily. I could now power up the Galileo by inserting the power cable

After completing the above steps I could now launch the Arduino IDE on my laptop. After launching the IDE I selected the Galileo: **Tools > Board > Intel Galileo**
then I selected the port: **Tools > Serial Port > dev**
I then completed the following steps to upgrade the firmware:

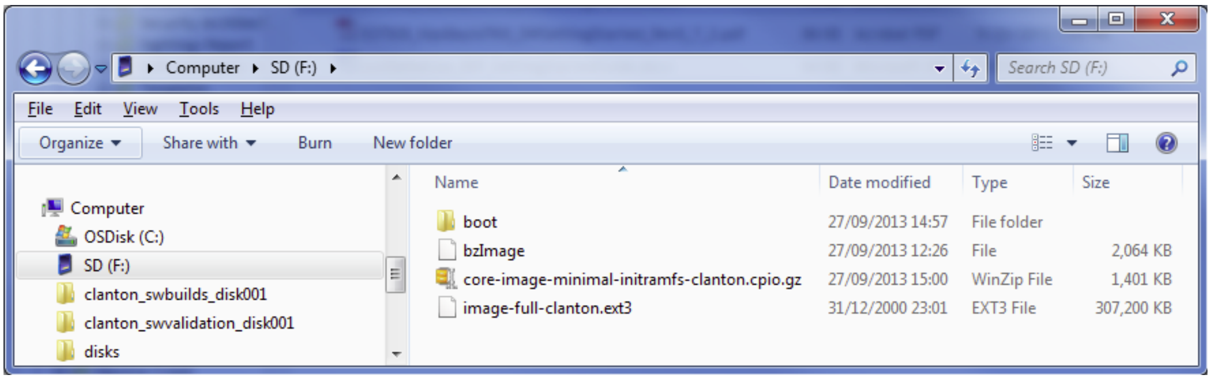
1. Launch the software upgrade using **Help > Firmware Upgrade**
2. A message is displayed asking you to confirm that the 5V power cable is plugged in. Click yes if it is connected. If no cable is plugged in, exit the upgrade process by selecting No, connect the power, and restart this process.
3. The board can be upgraded to newer software or downgraded to older software. The next message displays the current software version that is on the board and the software version that you are trying to flash onto the board. Select yes to either Upgrade/Downgrade or flash the same software again.
4. The upgrade progress takes about 6 minutes and is displayed in several popup messages. During the upgrade process, you will not have access to the IDE.

Note: The power and USB cables must stay connected during the upgrade process.

5. When the upgrade is finished a message is displayed stating the new firmware version, upgrade successful. Again I had to repeat these steps for all the boards so the all had the same firmware.

Booting board from the SD card (Mandatory for Wi-Fi)

To install the Linux image I had to download it from the following location: https://downloadcenter.intel.com/Detail_Desc.aspx?DwnldID=23171. Without installing the Linux image I would not be able to use Wi-Fi or any of the programming languages that come preinstalled on this Linux OS or even SSH into the Galileo. After doing that I then extracted the files over to the micro SD card so it had the same files as the image below:


Computer ▸ SD (F:) ▸
Search SD (F:)
File Edit View Tools Help
Organize ▾ Share with ▾ Burn New folder
Computer
OSDisk (C:)
SD (F:)
clanton_swbuilds_disk001
clanton_swvalidation_disk001
disks

Name	Date modified	Type	Size
boot	27/09/2013 14:57	File folder	
bzImage	27/09/2013 12:26	File	2,064 KB
core-image-minimal-initramfs-clanton.cpio.gz	27/09/2013 15:00	WinZip File	1,401 KB
image-full-clanton.ext3	31/12/2000 23:01	EXT3 File	307,200 KB

After doing that I removed the SD card from my laptop, powered off the Galileo and inserted the SD card into the slot on the Galileo. I then plugged the Galileo board back into the mains so it could boot the new Linux OS

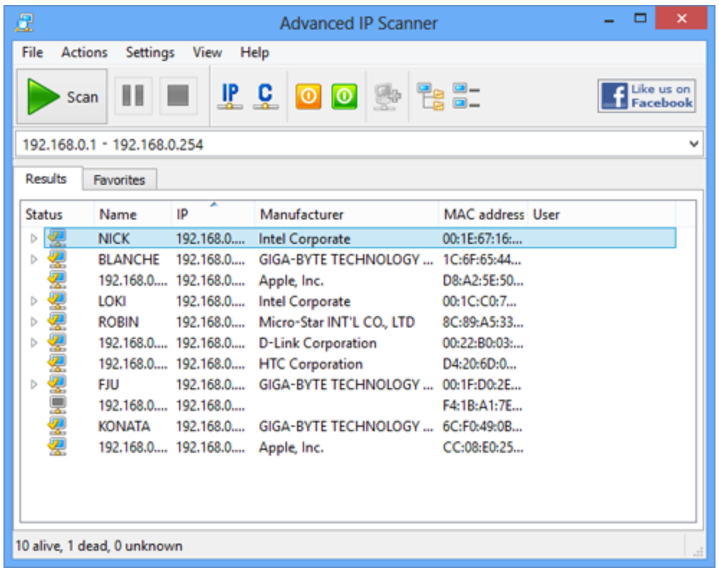
Note: It can take a couple of minutes for the Galileo to boot for the first time as it has to generate SSH keys

After completing the above steps Galileo was now successfully running a Yocto Linux OS. Again I had to repeat these steps on the other four boards.

Galileo Remote Login Via SSH

Before I could remotely login to one of the Galileo microcontrollers I first had to find the IP address of the device. Since the Galileo boards are headless (No monitor) and I didn't have access to the office router, this was not as straightforward as I first thought. To find the IP I did the following:

1. Download the following IP scanner software from the following link: http://www.advanced-ip-scanner.com/
2. Using an Ethernet cable plug one end into a laptop or PC with wireless capabilities and plug the other end into the Galileo microcontroller. The laptop or PC should now dynamically assign the connected Galileo an IP address.
3. Then run the IP scanner software downloaded in the first step and hit the scan button. The software will proceed to scan for local devices and lists their IP address. After a couple of minutes, you should see a device named: clanton.mshome.net with the manufacturer being Intel Corporation showing up in the list. You will also see the IP address that your computer dynamically assigned to this device. With this IP address, we can now remotely login to this device.



192.168.0.1 - 192.168.0.254

Status	Name	IP	Manufacturer	MAC address	User
	NICK	192.168.0....	Intel Corporate	00:1E:67:16:...	
	BLANCHE	192.168.0....	GIGA-BYTE TECHNOLOGY ...	1C:6F:65:44...	
	192.168.0....	192.168.0....	Apple, Inc.	D8:A2:5E:50...	
	LOKI	192.168.0....	Intel Corporate	00:1C:C0:7...	
	ROBIN	192.168.0....	Micro-Star INT'L CO., LTD	8C:89:A5:33...	
	192.168.0....	192.168.0....	D-Link Corporation	00:22:B0:03:...	
	192.168.0....	192.168.0....	HTC Corporation	D4:20:6D:0...	
	FJU	192.168.0....	GIGA-BYTE TECHNOLOGY ...	00:1F:D0:2E...	
	192.168.0....	192.168.0....		F4:1B:A1:7E...	
	KONATA	192.168.0....	GIGA-BYTE TECHNOLOGY ...	6C:F0:49:0B...	
	192.168.0....	192.168.0....	Apple, Inc.	CC:08:E0:25...	

10 alive, 1 dead, 0 unknown

Figure 5: Advanced IP Scanner Interface

4. Once we have to IP address we can use the Putty application to remotely SSH into the Galileo microcontroller, putty can be downloaded here: http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html
5. After running putty enter the IP into the Host Name or IP textbox and click open.
6. Putty will now open a terminal connected to the Galileo using SSH. The Galileo will then ask for a username:

   Note: The default username is: root and no password is required

   Enter this and press enter, you should now be logged into the Galileo microcontroller over SSH.

By typing "ls" into the terminal you will be returned a list of files that are essential for this project. Each file has its own purpose and will be explained in the rest of the report
Note: This file should not be deleted

Setting up AD-HOC Wireless Network

To configure ad-hoc networking on the Galileo device, I first had to do it manually by entering the commands through the Linux command line using the terminal we opened in the previous section. The commands were as follows:

1. First, bring down the wlan0 interface:
   ```
   ip link set wlan0 down
   ```
2. Set wlan0 to use AD-HOC mode:
   ```
   iwconfig wlan0 mode ad-hoc
   ```
3. Set the wireless channel we want the Galileo microcontrollers to communicate over:
   ```
   iwconfig wlan0 channel 4
   ```
4. Give the ad-hoc network a name (SSID):
   ```
   iwconfig wlan0 essid 'GalileoAD-HOC'
   ```
5. Set the Galileo to use the following MAC address:
   ```
   iwconfig wlan0 ap AA:BB:CC:DD:EE:FF
   ```
6. Give the ad-hoc network a WEP wireless encryption key:
   ```
   iwconfig wlan0 key 1234567890
   ```
7. Bring back up the wlan0 interface:
   ```
   ip link set wlan0 up
   ```
8. Give the Galileo device a static IP address:
   ```
   ip addr add 192.168.1.221/24 dev wlan0
   ```

Note: The only change made to these steps for each device is we give each one a different static IP address

These were the steps used to set up and broadcast an ad-hoc Wi-Fi beacon on the first Galileo. To speed up this process I programmed a bash script (ad_hoc_numberOfDevice.sh) for each Galileo. When run, the script will automatically configure the device to produce an ad-hoc Wi-Fi beacon.

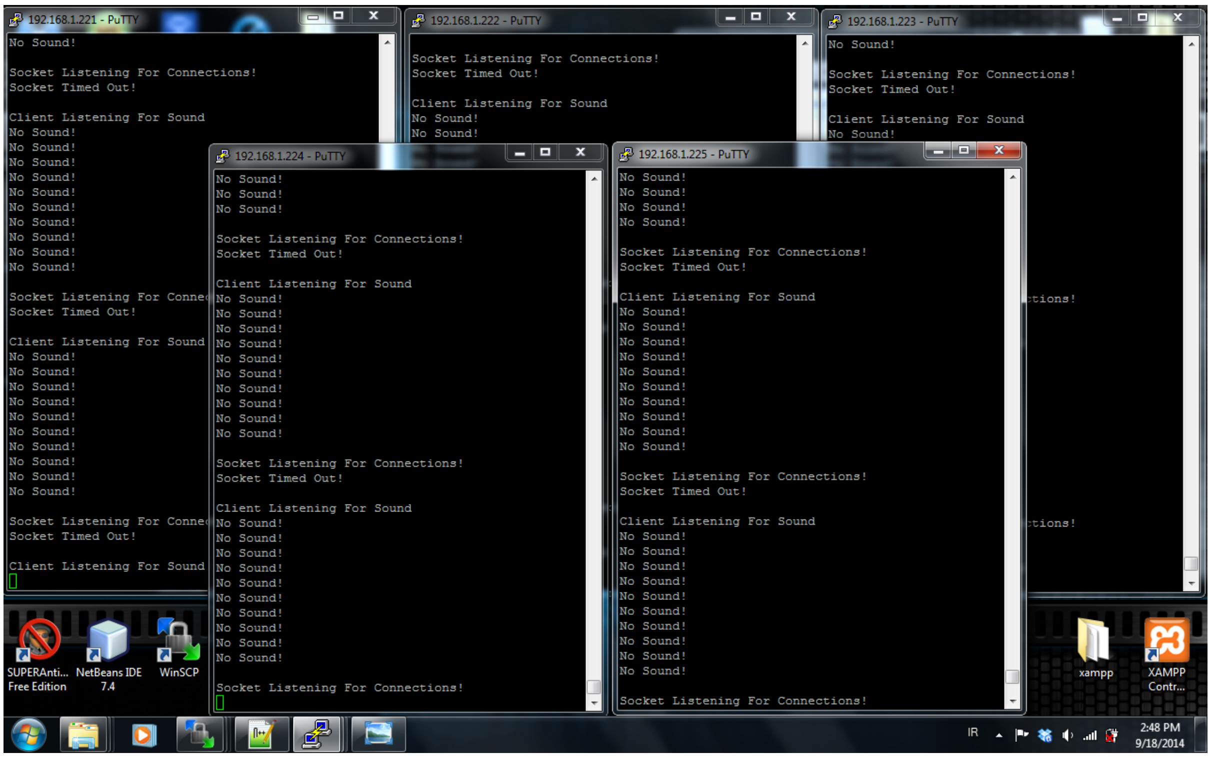
So when you log in to the device with putty after finding the dynamic IP assigned by your computer you can simply type **sh ad-hoc_1.sh** (for the first device) to run the script, the script will then complete the eight steps above automatically. After doing this on the first device I had one device that was now broadcasting an ad-hoc Wi-Fi beacon. The next thing was to complete the steps used in the **Galileo remote Login via SSH** and set the **AD-HOC Wireless Network section** on every device. In short, just find the IP of the device, log in with putty and run the ad-hoc script for all five devices. Once this is complete all devices will have the same SSID, MAC Address and WEP encryption key but different IP address. Each device is also broadcasting the same ad-hoc Wi-Fi beacon which they will use to communicate over.

The next step was to actually connect my computer to this ad-hoc network. This would allow my computer to communicate with all the Galileo microcontrollers over the same ad-hoc network. I did this by using the following steps:

1. Locate the list of wireless networks on the computer.
2. From this list select the GalileoAD-HOC network (Has an image of three computers connected together) and clicked the connect button.
3. A popup box will then prompt you to enter the WEP security key:

   WEP Key: 1234567890

   Then click the OK button.
4. The computer will now be connected to the GalileoAD-HOC network.



192.168.1.221 - PuTTY
```
No Sound!

Socket Listening For Connections!
Socket Timed Out!

Client Listening For Sound
No Sound!
No Sound!
No Sound!
No Sound!
No Sound!
No Sound!
No Sound!
No Sound!
No Sound!
No Sound!

Socket Listening For Conne
Socket Timed Out!

Client Listening For Sound
No Sound!
No Sound!
No Sound!
No Sound!
No Sound!
No Sound!
No Sound!
No Sound!
No Sound!
No Sound!

Socket Listening For Conne
Socket Timed Out!

Client Listening For Sound
```

192.168.1.222 - PuTTY
```

Socket Listening For Connections!
Socket Timed Out!

Client Listening For Sound
No Sound!
No Sound!
```

192.168.1.223 - PuTTY
```
No Sound!

Socket Listening For Connections!
Socket Timed Out!

Client Listening For Sound
No Sound!
```
```
ctions!
```
```
ctions!
```

192.168.1.224 - PuTTY
```
No Sound!
No Sound!
No Sound!

Socket Listening For Connections!
Socket Timed Out!

Client Listening For Sound
No Sound!
No Sound!
No Sound!
No Sound!
No Sound!
No Sound!
No Sound!
No Sound!
No Sound!
No Sound!

Socket Listening For Connections!
Socket Timed Out!

Client Listening For Sound
No Sound!
No Sound!
No Sound!
No Sound!
No Sound!
No Sound!
No Sound!
No Sound!
No Sound!
No Sound!

Socket Listening For Connections!
```

192.168.1.225 - PuTTY
```
No Sound!
No Sound!
No Sound!
No Sound!

Socket Listening For Connections!
Socket Timed Out!

Client Listening For Sound
No Sound!
No Sound!
No Sound!
No Sound!
No Sound!
No Sound!
No Sound!
No Sound!
No Sound!
No Sound!

Socket Listening For Connections!
Socket Timed Out!

Client Listening For Sound
No Sound!
No Sound!
No Sound!
No Sound!
No Sound!
No Sound!
No Sound!
No Sound!
No Sound!
No Sound!

Socket Listening For Connections!
```

Figure 6: My computer logged into all five Galileo Microcontrollers

With my computer now connected to GalileoAD-HOC network I again needed to remotely login to each Galileo device using putty but this time I used the static IP addresses I gave to each device (192.168.1.221 - 225) shown in previous steps. Once this was complete I now had five terminals on my computer one connected to each Galileo microcontroller. This is shown in the image above and enabled me control all devices from my computer.

Troubleshooting

When trying to set up ad-hoc networking on each device don't use the network interface file to configure it, as it won't work, I tried changing a number of different settings in this file but never succeeded in accomplishing it. The advantages would have been huge as the steps in the two sections above would be done automatically when each device is powered on.

The computer I was using to connect to the GalileoAD-HOC network was running windows 7 which was the simplest way to actually connect to the ad-hoc network. When I tried a computer running windows 8 it could never pick up the ad-hoc Wi-Fi beacon the Galileo's were broadcasting. I had to manually locate and add the network to the list of wireless networks which can be time consuming. My advice here is to use a computer running windows 7 which will make things a lot easier when trying to connect to the GalileoAD-HOC network

Constructing the Demonstration Application

The proposed solution was to implement a network of devices which sense sounds and communicate with each other to make decisions and determine which device is closest to the sound. This could then be extended to localize and track sound sources or an interesting event in the environment. I extended the Galileo's by adding microphones to each unit and to indicate the conclusion of the decision process I also added LEDs to each board.

Hardware Components

1. Intel Galileo Microcontroller x1
2. Sound Sensor x1
3. Male to Male Jumper Cables x11
4. LEDs x3 (Red, Amber, Green)
5. 180omhs Resistors x3
6. Breadboard x1
7. Wireless Antenna x1

All hardware used to construct the demonstration application is shown in the image below. Most of the electronic equipment I purchased myself in the local Maplin electronics store but I had to purchase the sound sensors online as there was nothing like the sensor I purchased available locally. As stated at the start of the report Intel donated five of the Galileo microcontrollers for me to use.

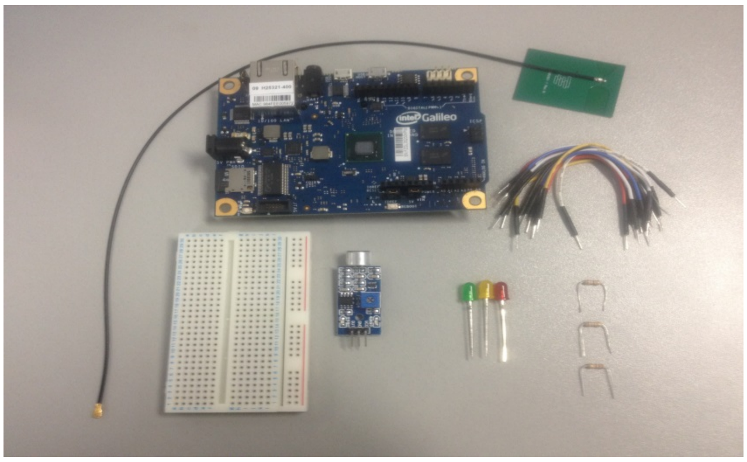


Figure 7: Electronic components use to construct the demonstration application

After assembling all the hardware components into an application, I had a prototype that looked like this:

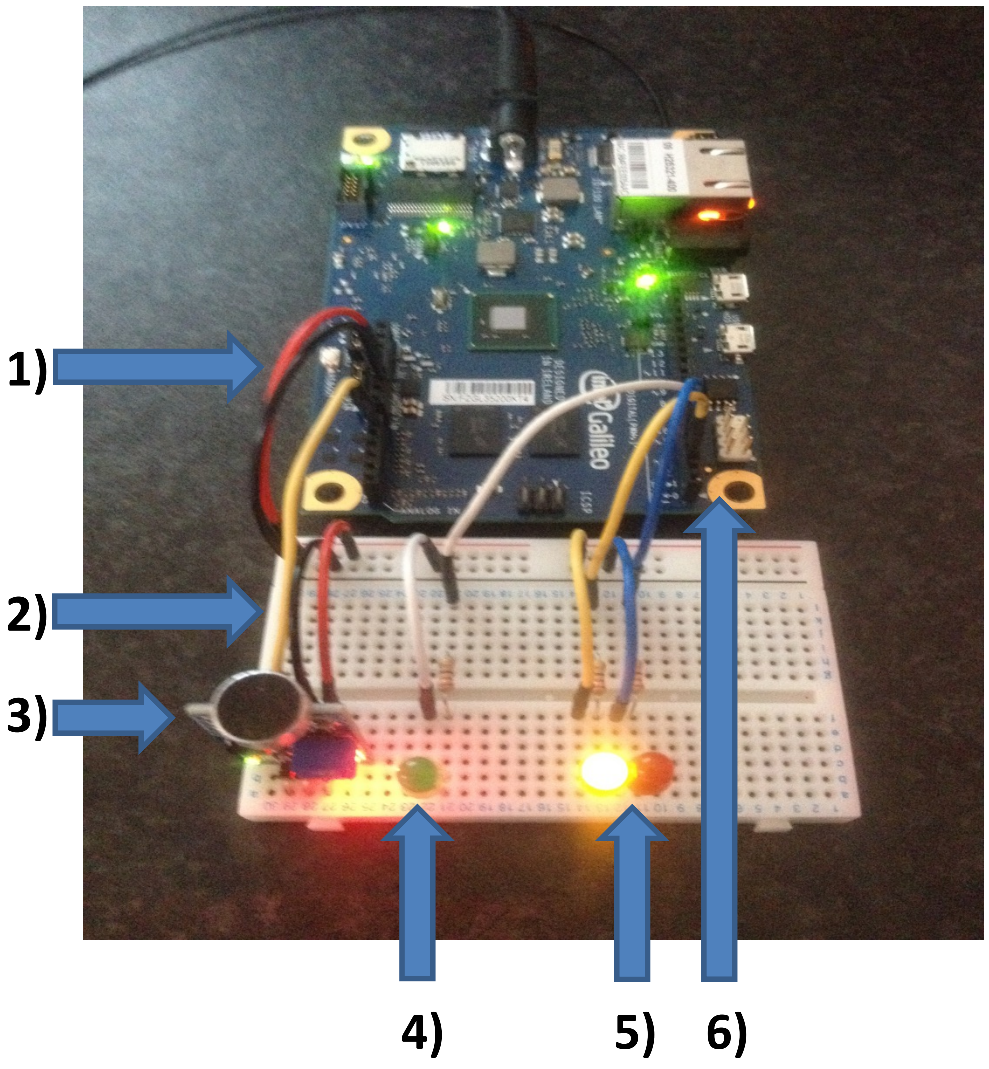
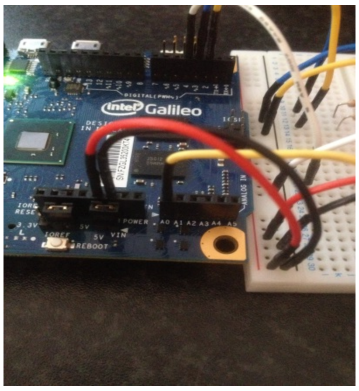


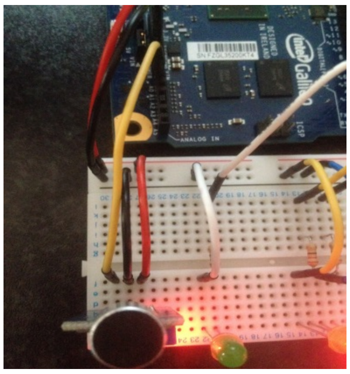
Figure 8: Fully assemble Galileo connected to its sound sensor and three LED's.

1. **The power and ground cables:**
   The red jumper cable is used to supply 5V power to the breadboard power rails. Plug the red jumper into the 5v output on the Galileo, as you'll see from the images there is a strip of eight outputs on the left of the Galileo in the middle of these pins you will see one marked 5v, plug the red jumper into that output. Now plug the other end into the breadboard making sure to plug it into the rail with the red strip running down the side. Now the breadboard has power running down the rail which we'll use to power the sound sensor and the LEDs.

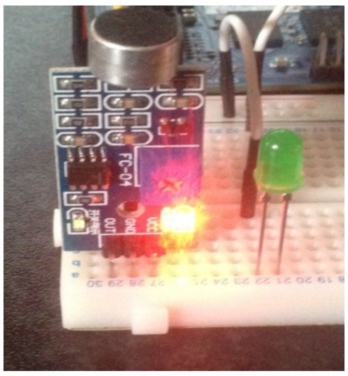


   The next thing was the ground for which I used a black jumper cable. This was plugged in next to the 5V power jumper cable as shown in the image right. The other end must be plugged into the ground rail on the breadboard, but also make sure it's plugged into the rail with the black line running down the side.

2. **The sound sensor analog output cable**
   To capture the data that the sound sensor was throwing out I connected a yellow jumper cable to the first analog input on the Galileo as shown right. The analog inputs are at the left of the Galileo and there are six in total. The other end is plugged into the strip on the breadboard that is also connected to the out from the sound sensor this is to transfer the data (varying voltage) to the Galileo.



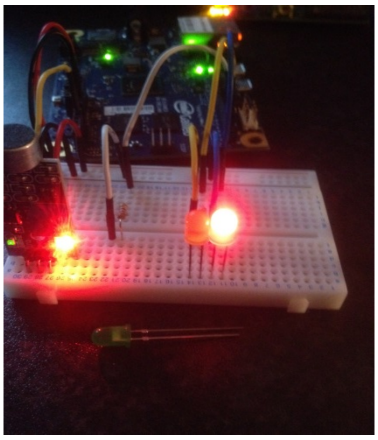
3. **Installing the Sound Sensor**
   The sound sensors I use were very basic but perfect for what I was trying to achieve. As you can see from the image on the right the sensor had three pins coming out from the end of the sensor. The pins were used as follows:



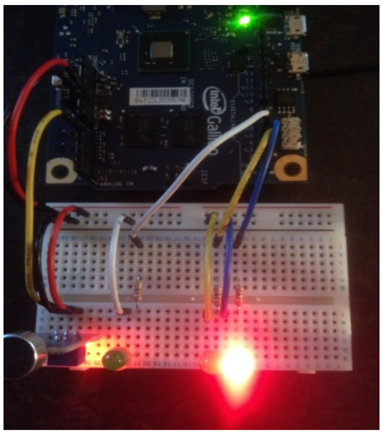
   1. **Analog output:** as I explained in the second step the output from the sensor is fed through the yellow jumper cable and into the first analog input on the Galileo.
   2. **Ground pin:** as you can see from the image above I used another black jumper cable to take ground from the ground rail and feed it into the sound sensor.
   3. **Power pin:** I also used a red jumper cable to take power from the power rails and feed it into the sound sensor

The sound sensor was now set up to sense sounds from the surrounding environment and feed the data back to Galileo. You may also notice a green LED next to the sensor this was used for the conclusion of the decision process.

4. **Setting up the LEDs**



For this project, I decided to go with three LEDs, The green LED as shown in the image above is used to indicate a sound has been detected above a certain threshold and to also show the conclusion of the decision process. The red and amber LEDs are used for the coordinated sequence protocol to indicate when the Galileo boards are communicating with each other. To set up the LEDs iii used three jumper cables and plugged them into the first three digital outputs on the Galileo these pins are number 2 – 4 and are on the right of the Galileo.



Amber = digital pin 2

Red = digital pin 3

Green = digital pin 4

I then plugged the other sides of the jumper cables into the breadboard as shown in the image right. I also had to use resistors to restrict the flow of electricity coming from the Galileo's digital output

Note: not using the resistors may melt the LEDs.

Then I used another three jumper cables to ground the LEDS to the ground rail. As you'll see from the images the green LED in front of the breadboard has two pins coming out from the bulb, one of these pins is longer than the other. The longer pin is always connected to the resistor which is in turn connected to the Galileo digital output pin.

5. **Running the coordinated protocol client**

To run the client you simply type the following command into the terminal for the first device:

```
Python client_1.py
```

This will run the client for the coordinated sequence protocol an begin an alternating loop where the node will listen for sounds for half a second and then switch to listening for incoming connections from other nodes for a tenth of a second.

Troubleshooting

When the client is run for the first time it will flick on and off all the LEDs so you can check to see all of them are functioning correctly. Sometimes the red LED my not light up properly and this is because it is connected the PWM (Pulse Width Modulation) pin which is pin three on the Galileo. To solve this problem I wrote a simple script called **Fade.py**. Run this script and you will notice the red LED begin to slowly illuminate, after it is turned off terminate the Fade.py program and again run the client. The red LED should now be function as intended

The steps describe above had to be completed for all the Galileo Microcontrollers and when I was finished I had a demonstration application that looked like this which could now be used by researchers at Insight to build and test new distributed algorithms and protocols

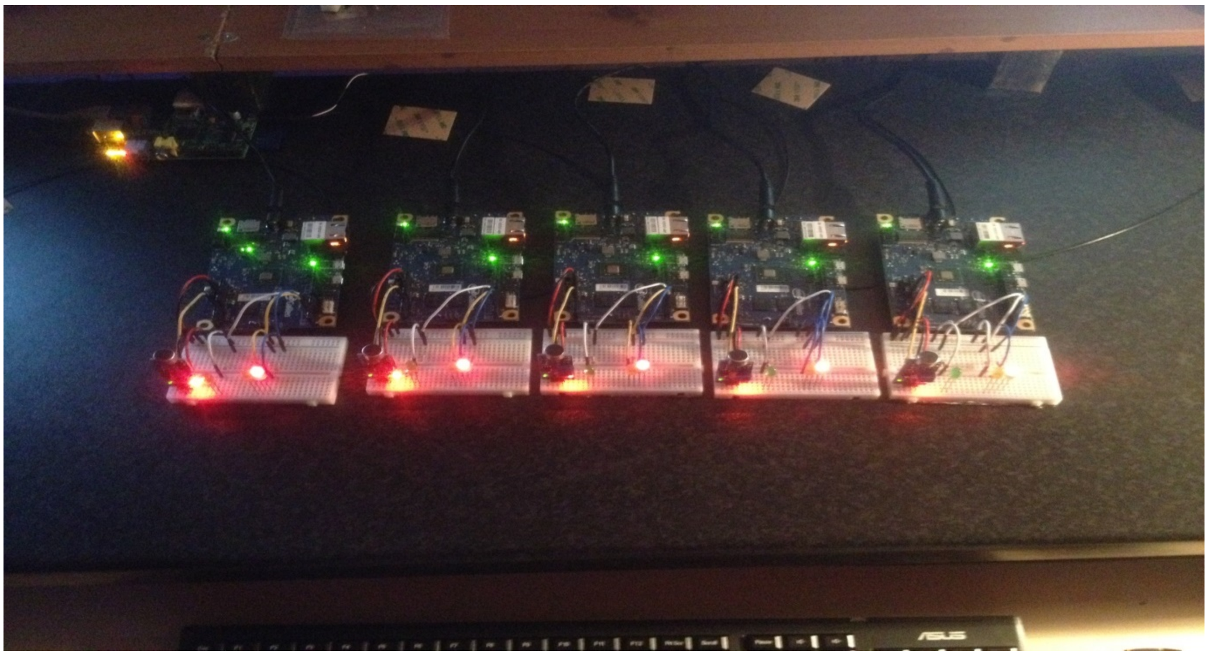


Figure 9: All five Galileo Microcontrollers connected to their sounds sensors and LED's. All devices are communicating using Wi-Fi over and AD-HOC wireless network.

The image above was taken when all the devices were running the coordinated protocol client which is why the red LED is illuminated indicating no sounds have been heard and there is no communication taking place between the connected devices

Coordinated Protocol

To make use of this wireless test bed I had to write a coordinated sequenced protocol for each Galileo. For the coordination protocol, any device that hears a sound above a threshold can initiate the process by broadcasting a message to all devices (usually, we assume a fully connected network). To visually indicate a device has detected a sound above the threshold the green LED will flash, and all devices will then begin a synchronous communication sequence in which each device will broadcast its sound level. The node with the lowest ID will broadcast its sound level first. Each node will then transmit in ID sequence until all nodes have broadcast their level. After the synchronous communication sequence was complete, each node could then decide who was the closest to the sound. I could then actuate and show the result by turning on a green LED for the node that had the highest sound level.

1. **Node Discovery**
   When the coordinated protocol is run for the first time I had to implement a way for each node to dynamically locate and identify the nodes operating in close proximity. To do this I made each node broadcast a “Who’s There” message to the network broadcast address 192.168.1.225 before listening for sounds. Every node that hears this message will store the IP address of the node that sent it into a hash table. The node that received the message will then reply back so the node who sent the “Who’s There” message in the first place can also store the IP address of the nodes around it. So when the first node runs the client it will broadcast its “Who’s there” and since it’s the only node running it will only get a reply from itself. I then run the second client on the next device and it will broadcast its “Who’s There” message. Node two will now get a reply from itself and node one so now both nodes are aware of each other. When I did this for all five nodes I now had a network of devices that are aware of and can communicate with all the nodes around them.

2. **Hash Table**
   The hash table is used again when we begin the sequence of sending sound levels. The hash table is used to store the sound levels of all devices so they can all decide who has the highest sound level.

3. **Resending Mechanism**
   In the event a node receives a packet out of sequence or a collision occurs and a packet is lost, I have implemented a resending mechanism which will broadcast a resent message to all nodes to indicate a packet has been received out of sequence

or lost. This message will contain the IP address of the node that has to resend the message so when that node receives the message it will resend its sound level again.

Conclusion

I successfully assembled an Intel Galileo wireless test best on which researchers can now try to test and implement new protocols and algorithms. I also constructed a demonstration application on top of our wireless test bed to demonstrate issues in multi-agent decision-making over wireless networks. When we have a small number of devices (three or four) and run the communication sequence at a reduced speed our protocol for sensing, communicating and actuation will work reliably but when we increase the number of devices and the speed at which they are communicating, packets arriving out of sequence and packet loss will increase dramatically. This can reduce the speed of the decision processes as we have to resend a lot of packets.

All the code I have written and everything I used to help me with my project during my time at Insight has been uploaded the Insight Git lab server and can be viewed at the following link http://uccgit.insight-centre.org/mbullman/intel_galileo_project/tree/master Also the entire project can be cloned from the Git Lab repository by using the following command:

git clone http://uccgit.insight-centre.org/mbullman/intel_galileo_project.git

Conference & Presentation & Abstract Paper

During my internship at Insight I was asked to do a presentation and to write an abstract research paper about the “Multi-Agent Decision Making over Wireless Networks” project I was working on. This was considered an essential part of my internship because for one this is what is expected of a full-time researcher and two it gave me experience and knowledge of how to write a research paper and also experience presenting in front of a room full of people. After completing my abstract paper I was asked to submit it to the Insight Student Conference 2014. My research paper was accepted for this conference so I then had to create a poster to show the work I had done during my Internship and this was displayed at the Insight student conference so that fellow interns and researchers at the Insight student conference could see and ask questions about my project. I have also uploaded the abstract paper, poster, and PowerPoint presentation I created to the Insight Git Lab server which can be viewed at the following link:
http://uccgit.insightcentre.org/mbullman/intel_galileo_project/tree/master

Further Research

To further research this project the next step would be to extend the multi-agent decision-making project to assume a network where all devices are not connected e.g. If we have a scenario in which we have three devices A, B, and C. Device A is in wireless range of device B but not in range of device C. Device C is also in range of device B but not device A. To explore this research the coordinated protocol will have to be extended to accommodate this non-connected network of devices. More research could also be conducted in extending the application to track sounds through the environment. With the protocol that's already in place, this can be achieved relativity easily. One would simply have to implement a mechanism to keep track of which node is closest to the sound. By doing this each node could form a path by using the geographic location of each node in the environment that has the highest sound level and use that to track and memorise the movement of the sounds.

Personal Experience

Before I started the project I had no idea of the challenges I would encounter during my internship The Internship has given me a wealth of experience and a real insight into the career of a researcher. It is it definitely a career path I will consider after completing my undergraduate degree. It was my first time being employed in the Information Technology sector and I was really grateful to Insight for giving me the opportunity. Before I began my internship I had little experience programming in the Python programming language or assembling/programming electronic hardware.

During this internship, I learned how to program in a new language (Python) and I gained vast experience in problem solving and working with wireless ad-hoc networks. I had to come up with and implement solutions to various problems on a daily basis. The project was one of the most difficult projects I have worked on to date. Working among fellow interns, PhD students and Post-Doc researchers on a day-to-day basis was also a brilliant experience as all were easy to approach and ask questions. This helped me to settle in really quickly and greatly improve my interpersonal and communication skills.

I can honestly say my twelve-week internship at Insight Centre for Data Analytics was a brilliant experience for me and has no doubt helped me to further my computer science career by giving me a chance to work in a real computer science environment. Writing the abstract paper, performing a presentation in front of colleges and presenting my poster at the Insight Student Conference 2014 was really beneficial to me as I had never done anything like this in the past. All in all, I will look back on my time at Insight with great enjoyment, I got to meet loads of new people who have a passion for the same things I do. It was truly a brilliant experience and one I'll never forget