Feasibility and Efficiency of Raspberry Pi as the Single Board Computer Sensor Node

Mohammad Rafiuzzaman
Department of Electrical and
Computer Engineering
The University of British Columbia,
Vancouver, BC, Canada

ABSTRACT

Low rate, low power utilization, and ease correspondence are one of the key focuses for the improvement of a practical and effective Sensor Network (SN) framework. This paper introduces this type of cost-effective and efficient sensor system with MFRC522 as sensors and Raspberry Pi as sensor nodes. Raspberry Pi brings the upsides of a Personal Computer (PC) to the space of SNs. This trademark makes it the ideal stage for interfacing with the wide assortment of outer peripherals as appeared in this research work. An efficient customized configuration process for both MFRC522 sensor and Raspberry Pi has been presented in this work in details. Other than this, a comparison of the key components and performances of Raspberry Pi with a portion of the current existing remote sensor nodes is also been presented in this work. This comparison demonstrates that regardless of few drawbacks, the Raspberry Pi remains an economical PC with its effective use in SN space and assorted scope of research applications.

Keywords

GPIO, MFRC522, Raspberry Pi, Raspbian, RFID, single board computer (SBC), sensor network

1. INTRODUCTION

Sensor Network (SN) offers a remarkable method for extricating information from unsafe land districts where human mediation is greatly troublesome. The perceptions acquired from SNs are exceptionally useful in numerous product applications like ecological, mechanical and meteorological observing, building and home mechanization, prescription, canny transportation, security, military protection, and so on [1]. With the complexity of different correspondence conventions and quick progressions of MicroElectro-Mechanical Systems (MEMS) innovations [2], sensors and SNs has turned into a developing region of research in scholastic, modern, and protection divisions. Diverse types of sensing technologies combined with processing power and network communication capability make SN very lucrative for their abundant use in near future.

The main building block of this SN is sensor node. An SN is made from spatially conveyed sensor hubs furnished with detecting gadgets to screen and to quantify qualities of the physical environment at various areas. For the most part, an ideal approach to actualizing a sensor hub is to associate the sensor with a Single Board Computer (SBC). As of now, in the market, there are numerous industrially accessible sensor hub stages to be utilized as SBC. In this paper, we proposed Raspberry Pi [3], a shoddy, adaptable, completely adjustable and programmable little PC board to be utilized as sensor hubs SBC.

One of the primary purpose for this decision was that, the Raspberry Pi conveys the benefits of a PC to the space of SN, which decreases the issues of deficient memory as accessible in another sensor hub SBCs [4, 5] and what makes it the ideal stage for interfacing with wide assortment of outside peripherals. Comparative analysis of the key components and performances of Raspberry Pi with a portion of the current existing remote sensor hubs like TelosB, Micaz, Irıs, Cricket, Lotus [4, 5] is displayed in this paper. Likewise, in this work, the execution of Raspberry Pi was contrasted and the most recent SBC like PandaBoard [6] and BeagleBone [7]. These examine demonstrate that, despite few inconveniences, the Raspberry Pi remains a reasonable PC with its effective use in SN area and various scope of research applications.

Other than these, this research work additionally portrays in points of interest the segments, outline, and working of a sensor hub with Raspberry Pi which has been produced, tried and confirmed for the advancement of a RFID entryway security framework. As RFID sensor, MFRC522-a profoundly coordinated reader/writer IC for contactless correspondence has been utilized.

2. RESEARCH MATERIALS:

Sensor nodes, as building blocks of SN, are consisted of four basic elements - The sensor unit comprised of a sensor, Processing unit comprised of processor & memory, the Transceiver, and the Power Unit as shown in Figure 1. Number and sorts of sensor nodes rely on upon the applications. Sensor nodes gather and exchange information utilizing four phases [4]:

1. Collecting the information,
2. Processing the information,
3. Packaging the information and
4. Communicating the information.

The outline and usage of a sensor node in this work have been done considering two noteworthy segments: SBC and MFRC522 RFID Card Reader/Writer. Backup segments have additionally been utilized to set up the protected sensor node.

2.1.SBC

The decision of the SBC relies on upon the sort of utilization which the sensor node employments. Here, this SBC was used to design an automatic RFID door security system, which can be controlled from a remote server. Raspberry Pi has been decided for this application as SBC because among different SBCs, Raspberry Pi is the least expensive SBC accessible with the best execution/cost and RAM/cost proportion. It's

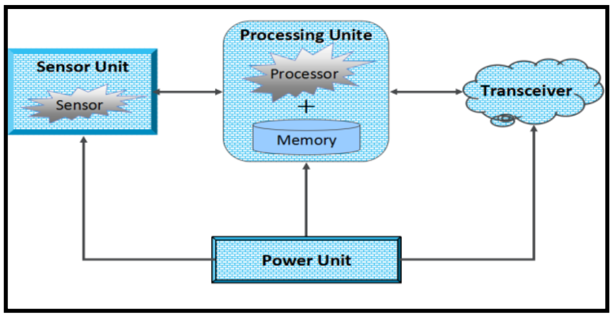


Figure 1. Typical sensor node diagram.

little size, ease, low power utilization and high preparing power make it reasonable for the plan of this body sensor.

2.1.1. Raspberry Pi:

Raspberry Pi is a little, intense, shoddy, hackable and training focused PC board presented in 2012 (Figure 2). This credit card measured PC with numerous exhibitions and moderate for 25-35$ is the ideal stage for interfacing with numerous gadgets.

The Raspberry Pi board contains a processor and illustrations chip, the program memory (RAM) and different interfaces and connectors for outside gadgets (Figure 2). Some of these gadgets are crucial, others are discretionary however all Raspberry Pi models highlight a framework on-a-chip setup worked around the Broadcom BCM2835 processor (a little yet genuinely capable portable processor regularly utilized as a part of mobile phones) that incorporates a CPU, GPU, sound/video preparing, and other usefulness all on a low-control chip; which is shabby, capable, and it doesn't expend a considerable measure of power [8]. As for inputs, Raspberry Pi can read the status of buttons, switches, and dials, or it can read sensors like temperature, light, motion, or proximity sensors (among many others) [9, 10].

2.2.MFRC522 RFID card reader/writer

The MFRC522 is an exceptionally incorporated reader/writer IC for contactless correspondence at 13.56 MHz. The MFRC522 reader underpins ISO/IEC 14443 A/MIFARE mode. As portrayed in Figure 3 the Analog Interface handles the modulation and demodulation of the analog signals. The Contactless UART deals with the convention prerequisites for the correspondence conventions in participation with the host. The FIFO Buffer support guarantees quick and helpful information exchange to and from the host and the Contactless UART and the other way around. Different host interfaces are actualized to meet distinctive client necessities. For the research interest of this work, Raspberry Pi has been used as a host interface. GPIO pins of Raspberry Pi are used to connect the SPI pins of MFRC522 per the connection diagram [11] shown in Figure 4. The MFRC522 supports contactless communication and uses bi-directional MIFARE higher transfer speeds up to 848 kBd [12].

2.3. Subsidiary components

A 5v 700mAh+ stable power source: The Raspberry Pi draws its energy from a microUSB port and requires a microUSB charger. Since the Pi is a smaller scale PC and not just a phone getting a battery finished off, an excellent charger with stable power conveyance that gives a steady 5v no less than 700mAh of yield must be utilized. Utilizing a low-quality/underpowered charger is the main wellspring of framework insecurity issues and disappointment with the Raspberry Pi. For this experiment, a USB mobile charger has been used.

An SD card (4GB+): Per the Raspberry Pi Foundation the base card suggested is a 4GB Class 4 card (the class speed demonstrates how quick it can read/compose). Since SD cards are modest nowadays, a 16GB Class 10 card was utilized as a part of this work.

HDMI cable: For computerized video/sound yield to an HDTV or screen or projector with HDMI bolster, an HDMI link is required. A projector connected to the pi with an HDMI cable has been used for the display.

Mouse/Keyboard: Any standard wired USB keyboard and mouse ought to work with no issues with the Raspberry Pi. Per the USB plan details USB-based keyboards and mouses ought to draw under 100mAh of power, however, numerous models neglect that determination and draw more. On the off chance that one discovers that his peripherals are drawing more than 100mAh every, he should utilize a controlled USB center point. A single USB Bluetooth mouse and keyboard was used for the Pi configuration.

Ethernet cable/Wi-Fi adapter: A Raspberry Pi can be shared to a hardwired LAN by means of Ethernet or can be associated with one of the numerous miniaturized scale Wi-Fi connectors which are good with the Pi. In this work, CAT-5 Ethernet cables were used for connecting the SBCs together.

12VDC 8.5A Switching Power Supply (100 Watt): It's a closed Frame Switching Power Supply used individually for supplying power to each individual sensor nodes.

With these components, Figure 5 depicts the stream of information inside the protected sensor node furthermore gives a clear delineation of the associations in it.

3. METHODOLOGY

The configuration of a single sensor node in details is presented in this section. The configuration is as follows:

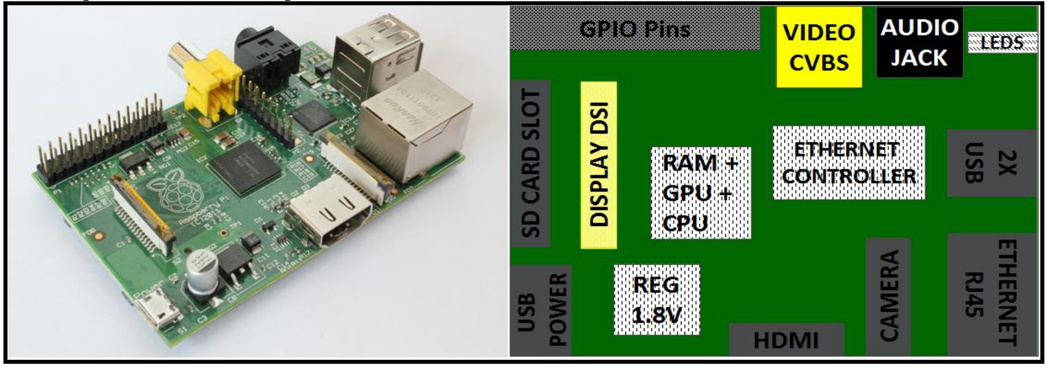


Figure 2. Raspberry Pi Model B and its internals

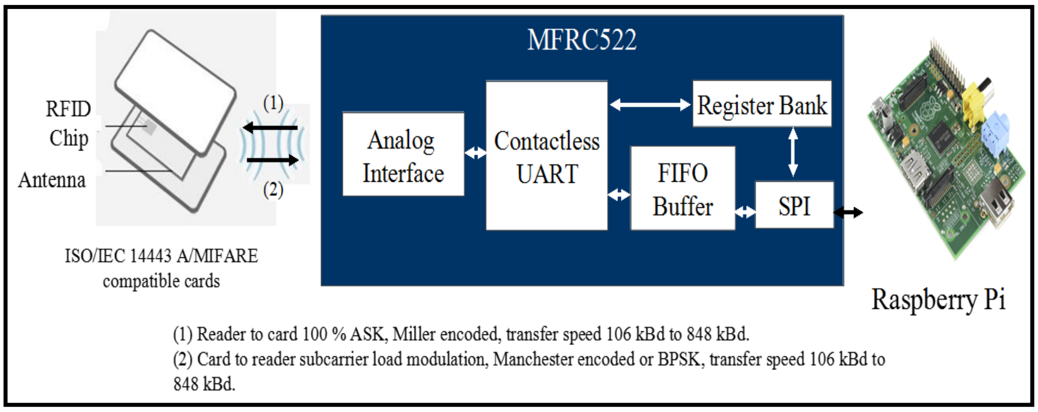

RFID
Chip
Antenna
(1)
(2)
ISO/IEC 14443 A/MIFARE compatible cards
MFRC522
Analog Interface
Contactless UART
Register Bank
FIFO Buffer
SPI
Raspberry Pi
(1) Reader to card 100 % ASK, Miller encoded, transfer speed 106 kBd to 848 kBd.
(2) Card to reader subcarrier load modulation, Manchester encoded or BPSK, transfer speed 106 kBd to 848 kBd.

Figure 3. MFRC522 Block Diagram with Read/Write Mode Communication

3.1.Selecting the Distribution

Like whatever other PC, the Raspberry Pi likewise utilizes a working framework and the "stock" OS is a kind of Linux called Raspbian [13], which have been utilized while outlining the arrangement of this work. Linux, as a free and open source program, is an incredible match for Raspberry Pi. On one hand, it keeps the cost of the stage low, and on the other, it makes it more hackable. There are likewise a couple non-Linux OS choices accessible [9]. The extra equipment and programming necessities can be accomplished by effectively existing equipment modules and open source programming.

Once the Raspberry Pi begins and runs it will be a totally free machine, however with a specific end goal to begin the procedure, another PC is expected to make the working framework picture on the SD card. The image which has been in this work was Raspbian “Wheezy” [14].

3.2.Selecting the Disk Imaging tool

After finishing downloading the disk image, it was extracted onto the SD card. For this experiment, Windows machine was utilized to remove the picture yet it's anything but difficult to play out a similar activity on Linux and OS X machines.

3.3.Imaging the SD Card

For this purpose, Win32 Disk Imager [14] was used. This program is intended to compose a crude disk image to a removable gadget or reinforcement a removable gadget to a crude disk image. It is exceptionally valuable for embedded improvement, to be specific Arm advancement ventures (Android, Ubuntu on Arm, and so forth).

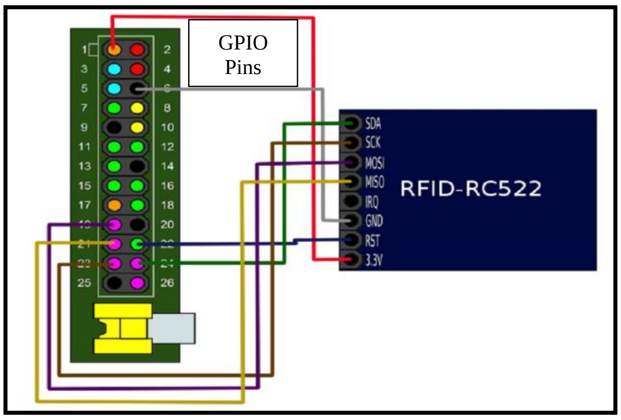


Figure 4. Connection diagram between Raspberry Pi and

At first, the SD card used for this project was inserted into the SD card reader of a computer. After that the .IMG file, which was downloaded was extracted from the Raspian .ZIP file. From within Win32 Disk Imager, the image file, and the appropriate drive letter was selected. Then the “Write” button was pressed to write the contents of the Raspbian image to the SD card. Once the application indicates the write has been successful the SD card from the computer was safely ejected.

3.4.Configuring Raspbian on Pi

After having an appropriately Flashed SD card, booting up the entire Pi at first was directed. Every one of the Cables and Peripherals was joined to the Raspberry Pi except for the Power Cable—this incorporates the HDMI or RCA link, the USB center, the Ethernet Cable/Wi-Fi Adapter, and so forth.

When every one of the Cables got joined to both the Pi and their separate goals, The SD Card was embedded. After the SD Card is situated immovably, the Micro USB Power Cable was embedded to start Booting up the Pi.

Very quickly the Boot succession goes looking over quickly by can be viewed as portrayed in Figure 6. After a minute or two, the Raspberry Pi will kick over into the Raspi-config.

Here some fundamental design errands were finished which upgraded the usefulness of the Pi unit. Even though the Raspi-config device can be keep running whenever it's optimal to do the heft of the tweaking and customization right from the

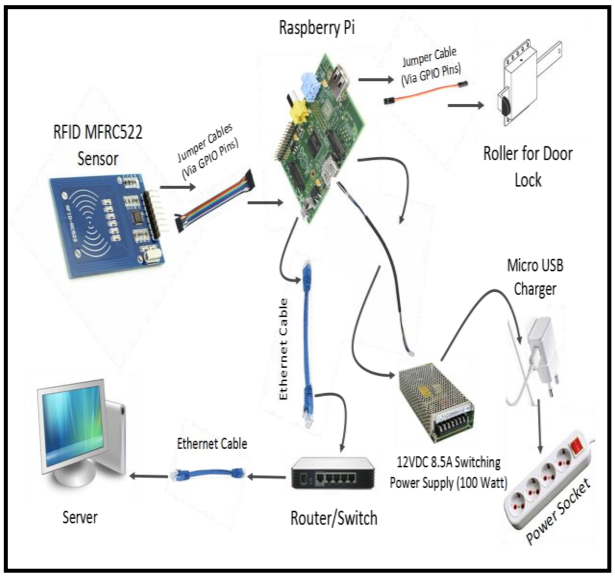
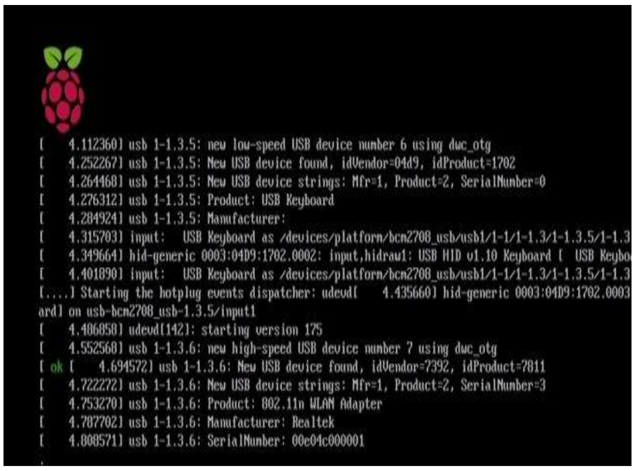


Figure 5. Flow diagram of a single sensor connection with the Server.



```
[    4.112360] usb 1-1.3.5: new low-speed USB device number 6 using dwc_otg
[    4.252267] usb 1-1.3.5: New USB device found, idVendor=04d9, idProduct=1702
[    4.264468] usb 1-1.3.5: New USB device strings: Mfr=1, Product=2, SerialNumber=0
[    4.276312] usb 1-1.3.5: Product: USB Keyboard
[    4.284924] usb 1-1.3.5: Manufacturer:
[    4.315703] input:   USB Keyboard as /devices/platform/bcm2708_usb/usb1/1-1/1-1.3/1-1.3.5/1-1.3
[    4.349664] hid-generic 0003:04D9:1702.0002: input,hidraw1: USB HID v1.10 Keyboard [  USB Keybo
[    4.401890] input:   USB Keyboard as /devices/platform/bcm2708_usb/usb1/1-1/1-1.3/1-1.3.5/1-1.3
[....] Starting the hotplug events dispatcher: udevd[    4.435660] hid-generic 0003:04D9:1702.0003
ard] on usb-bcm2708_usb-1.3.5/input1
[    4.486858] udevd[142]: starting version 175
[    4.552568] usb 1-1.3.6: new high-speed USB device number 7 using dwc_otg
[ ok [    4.694572] usb 1-1.3.6: New USB device found, idVendor=7392, idProduct=7811
[    4.722272] usb 1-1.3.6: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[    4.753270] usb 1-1.3.6: Product: 802.11n WLAN Adapter
[    4.787702] usb 1-1.3.6: Manufacturer: Realtek
[    4.808571] usb 1-1.3.6: SerialNumber: 00e04c000001
```

Figure 6. Boot screen of Raspberry Pi for the first time.

begin as later changes to the framework may strife with a portion of the designs errands.

3.5.Expanding the Rootfs:

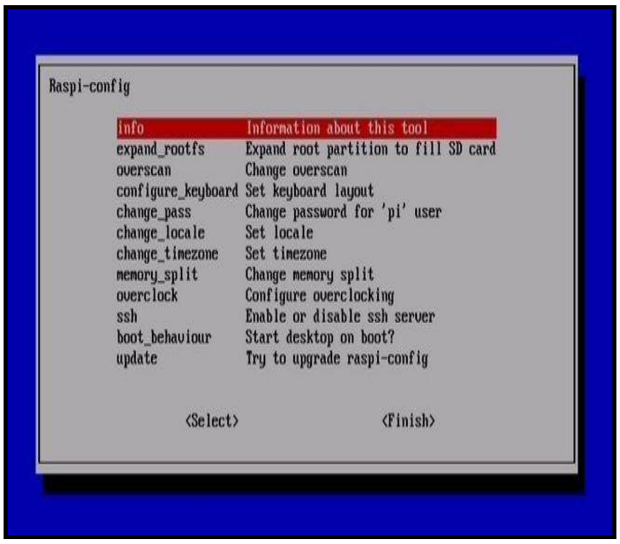
This is one of the most important configurations that had to be done to develop this system (second option on Figure 7). By default, Raspbian utilizes as quite a bit of its SD card as the core OS require. But for this work the entire SD card was needed to be accessed, so that plenty of storage for future projects would be available. For this reason, it is very important to enable this option. Without this, it was impossible for us to install JAVA eclipse IDE on the Pi, on which the development of most of this security protocol is carried out.

3.6.Overscan:

Typically, there is noteworthy dark space around the edges of the display (the Pi is utilizing the center bit of the screen and not the full screen) and overscan has been empowered so that the Pi can utilize the full screen of its display.

3.7.Updating the Software:

Before beginning the Pi, it's a smart thought to do an essential Software redesign. In the wake of setting up the system, testing the association, a framework wide Software overhaul must be done as such that all the most recent components of

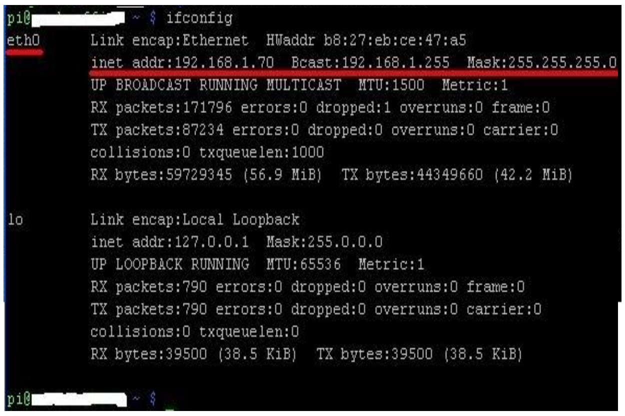


```
Raspi-config

info               Information about this tool
expand_rootfs      Expand root partition to fill SD card
overscan           Change overscan
configure_keyboard Set keyboard layout
change_pass        Change password for 'pi' user
change_locale      Set locale
change_timezone    Set timezone
memory_split       Change memory split
overclock          Configure overclocking
ssh                Enable or disable ssh server
boot_behaviour     Start desktop on boot?
update             Try to upgrade raspi-config

        <Select>                    <Finish>
```

Figure 7. Raspberry Pi configuration screen.



```
pi@[redacted] ~ $ ifconfig
eth0      Link encap:Ethernet  HWaddr b8:27:eb:ce:47:a5
          inet addr:192.168.1.70  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:171796 errors:0 dropped:1 overruns:0 frame:0
          TX packets:87234 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:59729345 (56.9 MiB)  TX bytes:44349660 (42.2 MiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:790 errors:0 dropped:0 overruns:0 frame:0
          TX packets:790 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:39500 (38.5 KiB)  TX bytes:39500 (38.5 KiB)

pi@[redacted] ~ $
```

Figure 8. Getting network information from router connected to

the IDE get empowered. For this purpose, the following commands were used:

sudo apt-get update

sudo apt-get upgrade

These commands train Raspbian to hunt accessible programming stores down for framework and programming redesigns and updates.

Installing JAVA: JAVA was installed on the Pi with these commands:

sudo apt-get install oracle-java7-jdk

After that the verification of this installation was carried out using this command:

sudo java -version

Next eclipse IDE was installed with the command bellow:

sudo apt-get install eclipse

The JDK version of this eclipse was needed to be configured after its installation.

3.8.Installing Python:

To utilize the module from Python, it is expected to stack an SPI wrapper, be that as it may, it was expected to introduce 'python-dev' to empower the establishment of the SPI wrapper. To introduce 'python-dev':

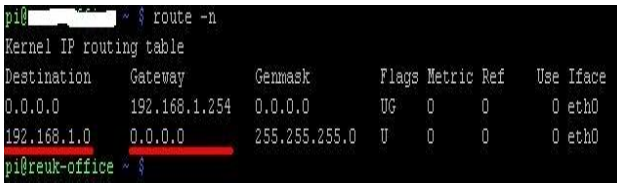
sudo apt-get install python-dev

command was used. After installing it, SPI was installed on the Pi using a customized built-in package.

3.9.Setting Up a Static IP on Pi

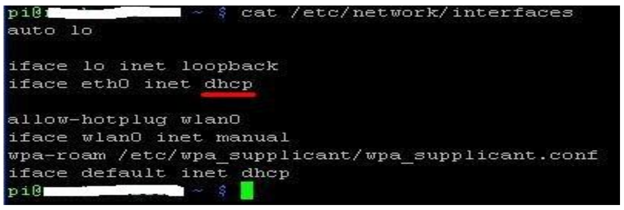
The Raspberry Pi is consequently set to get an IP address from the wired or remote system. In any case, Pi needs an IP address so that any traffic bound for the Raspberry Pi will have the capacity to discover it on the system.

This strategy for acquiring an IP address is called DHCP or Dynamic Host Configuration Protocol. It is here and there



```
pi@[redacted] ~ $ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.1.254   0.0.0.0         UG    0      0        0 eth0
192.168.1.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
pi@reuk-office ~ $
```

Figure 9. Getting network information from Raspberry Pi.



```
pi@[redacted] ~ $ cat /etc/network/interfaces
auto lo

iface lo inet loopback
iface eth0 inet dhcp

allow-hotplug wlan0
iface wlan0 inet manual
wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
iface default inet dhcp
pi@[redacted] ~ $
```

Figure 10. DHCP underlined showing dynamic IP assignment.

alluded to as a dynamic IP. Router will regularly disperse these IP addresses however it isn't ensured that it will get a similar IP address without fail. This can bring about issues if the Pi is should have been associated remotely. Hence, static IP address for the Pi is required.

Step 1. Checking the connection

To begin with, it is expected to twofold watch that the Raspberry Pi is associated with the network. For doing that the following command was used whose result is shown in Figure 8:

sudo ifconfig

As it can be found in Figure 8, the Raspberry Pi is associated with the network and has been allocated the IP address 192.168.1.70

Step 2. Keeping some information

Before applying a static IP deliver to the Raspberry Pi it is expected to accumulate the essential information from it. This information can be acquired from the "ifconfig" command which was run before. From a note of the accompanying information was made:

Current IP Address (inet addr) = 192.168.1.70

Broadcast Range (Bcast) = 192.168.1.255

Subnet Mask (Mask) = 255.255.255.0

With these noted down, the following command was run

sudo route –n

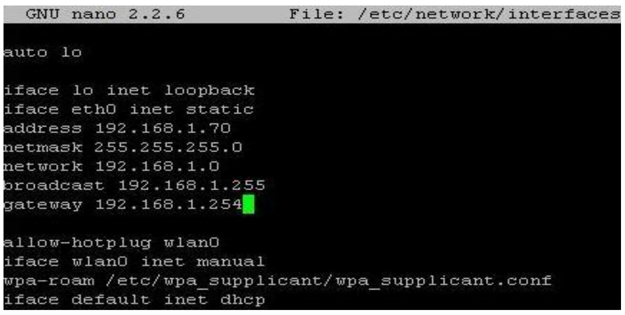
This gave the information from the router as depicted in Figure 9. From here the following information was taken:

Gateway = 192.168.1.254

Destination = 192.168.1.0

Step 3. Editing the files

The document expected to change to give the Raspberry Pi a static IP address is situated in/and so on/system/interfaces. This work used:



```
  GNU nano 2.2.6          File: /etc/network/interfaces

auto lo

iface lo inet loopback
iface eth0 inet static
address 192.168.1.70
netmask 255.255.255.0
network 192.168.1.0
broadcast 192.168.1.255
gateway 192.168.1.254

allow-hotplug wlan0
iface wlan0 inet manual
wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
iface default inet dhcp
```

Figure 11. Assigning static IP address to raspberry pi.

cat /etc/network/interfaces

command to see the important current settings as appeared in Figure 10. At that point the accompanying command was entered to alter the system interfaces record:

sudo nano /etc/network/interfaces

Above all else, the "dhcp" underlined in the screenshot before was supplanted with "static" which made the accompanying line:

iface eth0 inet static

At that point specifically underneath the accompanying order was entered utilizing the data which was noted down before.

address 192.168.1.70

gateway 192.168.1.254

netmask 255.255.255.0

network 192.168.1.0

broadcast 192.168.1.255

Here "network" is the subnet cover IP address. Here "address" is the IP address that is should have been relegated to the Raspberry Pi. In the wake of sparing the file, it looked like Figure 11.

Step 4. Reboot

After finishing all these configurations, the following command was used:

sudo reboot

to restart the Raspberry Pi with its new static IP address because the progressions that have been made will just produce results

after a reboot. To twofold watch that all is well, after login, the Gateway Address discovered before was pinged with the command: ping 192.168.1.254 -c 5

This pinged the gateway address and returned something like Figure 12.

3.10. Reading Data from MFRC522 Sensors

After configuring Raspberry Pi and enabling it for reading data from MFRC522 sensors information generation from MFRC522 sensors was started in the following way:

- In the beginning, SPI was enabled on the Raspbian to make it capable to read MFRC522.
- After that, the raspberry Pi runs a python script to enable MFRC522 related GPIO. Here the reading mode of MFRC522 was used only, not the writing mode.
- Later MFRC522 class library was imported and different definitions for it were set.
- After enabling every options when a card comes in contact with the MFRC522 it starts reading it and stores its ID value in a byte array as: string(user_id[0])+","+string(user_id[1])+","+string (user_id[2])+","+string(user_id[3]).
- If a successful read is conducted the “secured_protcol.jar” file of the system which is

written in JAVA is called from the running MFRC522 reading Python code.

- In the "secured_protcol.jar" file pi4j was used to set and read GPIO pins. Here the GPIO pins for this work was set as bellow:
    - GPIO_07 = doorPin: For controlling the roller lock.
    - GPIO_03 = ledSearchPin: For showing that the card is being read.
    - GPIO_04 = ledFoundPin: To show that the card is granted access.
    - GPIO_05 = lednotFoundPin: Showing card is not granted an access.
- After that the incoming user ID (UID) byte array was converted to Hexadecimal format in the "secured_protcol" program of this work for using it.
- The incoming UID with the UID database of this system were compared to find a match.
- If a match is found the "doorPin" is set to the state of low which cuts off the power of the roller lock for the door and the door opens.
- If the match is not found then the "doorPin" state remains high, which keeps the roller lock powered up and the door remains closed.

4. RESULT AND RESULT ANALYSIS:

The system proposed in this paper is both cost effective and efficient as Raspberry Pi was used as sensor nodes. To legitimize this announcement performance of Raspberry Pi with following sensor nodes [4, 5] were compared:

- TelosB - packages every one of the essentials for lab concentrates on into a solitary stage including USB programming capacity, an IEEE 802.15.4 radio with coordinated receiving wire, a low-control MCU with expanded memory and a discretionary sensor suite. This stage conveys low power utilization taking into consideration long battery life and additionally quick wake up from rest state.

- MicaZ – depends on the Atmel ATmega128L which is a low-control microcontroller and runs MoteWorks from its interior flash memory. The MICAz (MPR2400) IEEE 802.15.4 radio offers both rapid speed (250 kbps) and equipment security (AES-128).

- Iris – is utilized for empowering low-control WSN. Iris gives clients a wide assortment of custom detecting applications giving up to three times enhanced radio range and double the program memory over past eras of MICA Motes.

- Cricket - is an area mindful form of the well-known MICA2 low-power Processor/Radio module. The Cricket Mote incorporates most the standard MICA2 equipment and an ultrasound transmitter and recipient.

- Lotus - depends on the NXP LPC1758, 32-bit ARM Cortex-M3 based microcontroller. A solitary processor board can be designed to run sensor application/handling and the network/radio correspondences stack all the while. Lotus, as all beforehand specified sensor hub stages, has the information rate of 250 kb/s.

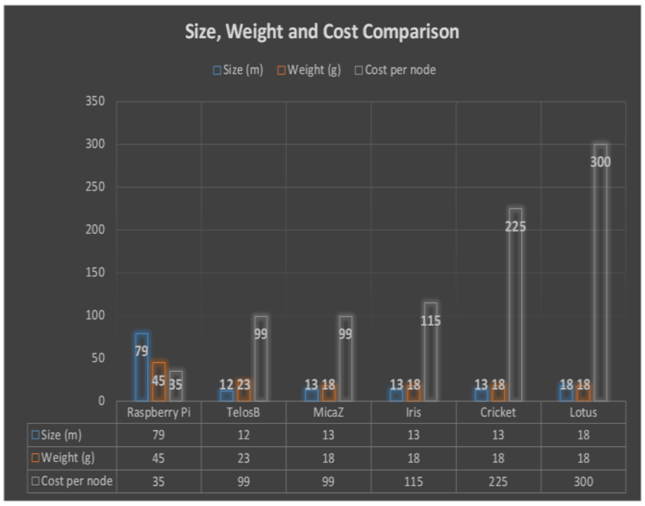
4.1 Size, Weight, and Cost Comparison

The physical size and cost of every individual sensor hub have a huge and direct effect on the straightforwardness and cost of deployment. Physical size effects the simplicity of system arrangement on the grounds that littler hubs can be set in more areas and utilized as a part of more situations. One of the primary objectives of each system is to gather information from whatever number areas as could be expected under the circumstances without surpassing altered spending plan. A diminishment in per-hub cost will bring about the capacity to buy more hubs, to convey a gathering system with higher thickness, and to gather more information [16]. The correlation of size, weight, and cost of fundamental models of Raspberry Pi or more specified remote sensor hubs is given in Figure 13.

By and large, the littler qualities are better. The qualities displayed in Figure 13 demonstrate that Raspberry Pi's favorable position against different frameworks lies in its littlest per unit cost.

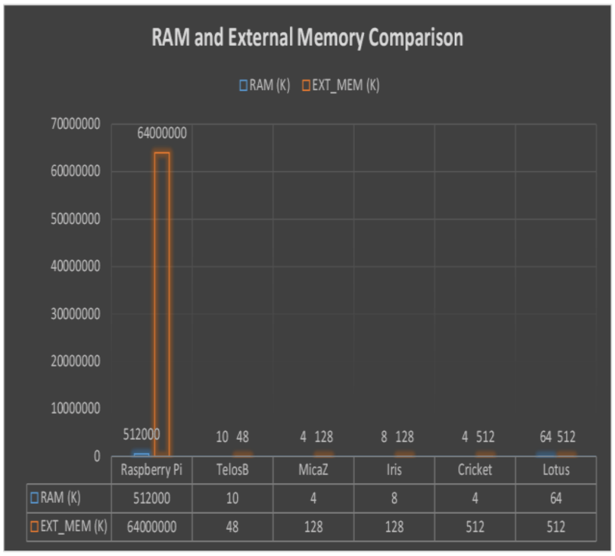
4.2 Memory Comparison

It is vital to note there's no hard drive on the Raspberry Pi; everything is put away on a Secure Digital (SD) Card. Albeit substantial SD cards holding 32 GB, 64 GB or more are accessible, they are frequently restrictively costly, however, the base required size of SD card is 2 GB relying upon the appropriation requests of the working framework.



	Raspberry Pi	TelosB	MicaZ	Iris	Cricket	Lotus
Size (m)	79	12	13	13	13	18
Weight (g)	45	23	18	18	18	18
Cost per node	35	99	99	115	225	300

Figure 13. Size, weight, cost comparison between Pi and other sensor nodes.



	Raspberry Pi	TelosB	MicaZ	Iris	Cricket	Lotus
RAM (K)	512000	10	4	8	4	64
EXT_MEM (K)	64000000	48	128	128	512	512

Figure 14. Memory comparison between Raspberry Pi and other sensor nodes.

It is likewise vital to note that the Raspberry Pi Model A has 256MB of RAM while the Model B has 512MB. Contrasted with other considered frameworks Raspberry Pi has the biggest measure of memory as appeared in Figure 14 what for the most part prompts to enhance general frameworks' performances.

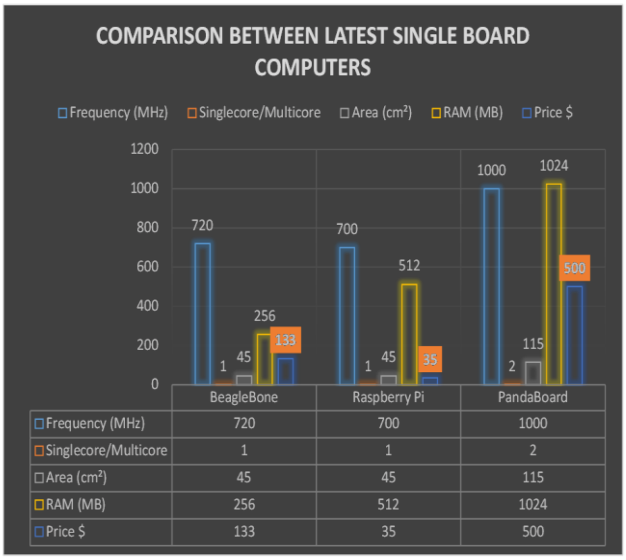
4.3 OS And Processor Comparison

Table 1 shows different processors and OS used in Raspberry Pi and other sensor nodes. The CPU is the primary part of the Raspberry Pi, in charge of doing the directions of a PC program through numerical and legitimate operations. The processor of Raspberry Pi is a 32 bit, 700 MHz System on a Chip (SoC), which is based on the ARM11 design and can be overclocked for more power [9]. ARM chips arrive in an assortment of designs with various centers arranged to give distinctive capacities at various value focuses. This implies most by far of the framework's parts – its focal and design handling units, sound and correspondences equipment alongside 512 MB memory chip, are incorporated with a solitary segment. The ARM-based BCM2835 is the motivation behind why the Raspberry Pi can work on simply the 5V 1A control supply gave by the onboard smaller scale USB port. The Raspberry Pi for working requires up to 700mA [14]. The unit of Raspberry Pi can be controlled utilizing a scope of force sources (expecting they can give enough current ~700mA).

Sensor hubs run inserted programming that specimens the physical environment, stack information, totals and communicates with a more elevated amount (peers or

Table 1. Processor and OS used in Raspberry Pi and other sensor nodes.

	Processor	Operating system
Raspberry Pi	ARM BCM2835	RASPBIAN
TelosB	TI MSP430	MOTE RUNNER, TINY OS
MicaZ	ATMEGA128	MOTE RUNNER, TINY OS
Iris	ATMEGA1281	MOTE RUNNER, TINY OS
Cricket	ATMEL128L	TINY OS
Lotus	ARM NXP LPC1758	RTOS, TINY OS



	BeagleBone	Raspberry Pi	PandaBoard
Frequency (MHz)	720	700	1000
Singlecore/Multicore	1	1	2
Area (cm²)	45	45	115
RAM (MB)	256	512	1024
Price $	133	35	500

Figure 15. Comparing Pi with the newest sensor nodes.

gateways). Despite the progressive approach, every sensor hub still needs a program, and the most widely recognized ways to deal with programming every sensor hub are to either program it by utilizing some type of a working framework or to pick a more elevated amount of reflection. The working frameworks change from customary working frameworks as far as objectives and procedure and every framework contrasts generously in the way to deal with memory security, dynamic reinventing, string model, constant components, and so on [16].

Like each PC, the Raspberry Pi needs a working framework, and the favored one for the Raspberry Pi is Linux dispersion. That is somewhat because it's free, yet chiefly this is because it keeps running on the Raspberry Pi's ARM processor [9]. There is a different dispersion of Linux ported to the Raspberry Pi's BCM2835 chip, including Debian, Fedora Remix, and Arch Linux [17]. The Raspberry Pi utilizes a working framework called Raspbian in view of Linux yet there are likewise a couple non-Linux OS alternatives accessible.

There are a few explanations behind choosing to run with the Raspbian working framework [17] rather than Tiny OS [18]:

- Raspbian has a desktop domain like Windows and Mac called Lightweight X11 Desktop Environment (LXDE), so it gives a simple move to those not acquainted with Linux command line.
- It comes pre-introduced with programming valuable for composing codes.
- The OS has been custom fitted to keep running on the Raspberry Pi. The code aggregation is enhanced for on-chip gliding point computations (hard-skim) as opposed to a slower programming based technique.
- There is across the board group bolster for the OS.

4.4 Communication Comparison

A key assessment metric for any SN is its communication rate, power utilization, and range. The communication rate likewise significantly affects hub execution. Higher correspondence rates mean fewer transmissions time and lower network power utilization. In any case, an expansion in

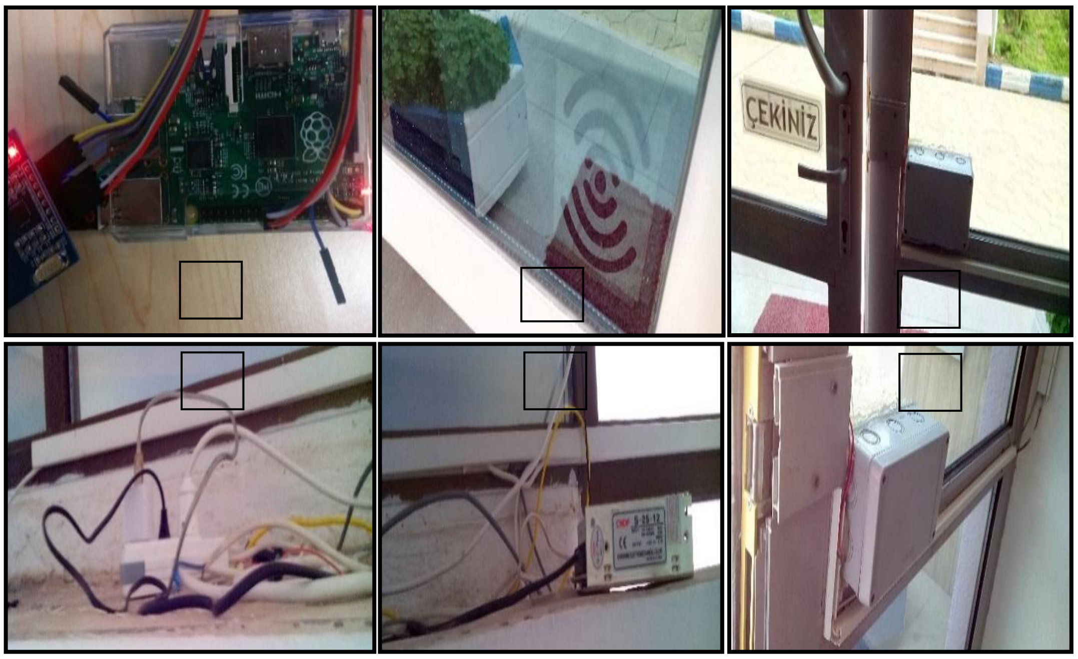


Figure 16. Actual implemented system of this research work (a single section only). Section A shows the connection of a single SN unit with a Raspberry Pi and an MFRC522 sensor. Section B, C and F shows one of these units mounted in a door. Section D and E shows the power and internet connection to this unit.

radio piece rate is regularly joined by an increment in radio power utilization. Considering present circumstances, a higher transmission bit rate will bring about higher framework execution. Be that as it may, an expansion in the correspondence bit rate significantly affects the power utilization and computational prerequisite of the hub [17].

The Ethernet port is the Raspberry Pi's primary gateway for correspondence with different gadgets and the Internet. The model B has a standard RJ45 Ethernet port while model A doesn't, however, can be associated with a wired system by a USB Ethernet connector. The Raspberry Pi's Ethernet port is auto-detecting which implies that it might be associated with a switch or specifically to another PC (without the requirement for a hybrid link) [3, 9]. USB Ethernet connector has a two-speed mode, 10 Mb/s and 100 Mb/s. With a link associated, the Raspberry Pi will naturally get the subtle elements it needs to get to the Internet when it stacks its working framework through the Dynamic Host Configuration Protocol (DHCP). This relegates the Raspberry Pi an Internet Protocol (IP) address on the system and lets it know the passage it needs to use to get to the Internet (ordinarily the IP address of switch or modem). Web availability of Raspberry Pi other than an Ethernet/LAN link (standard RJ45 connector) might be by means of a USB Wi-Fi connector.

5. FURTHER DISCUSSION AND RESULT ANALYSIS:

Besides the comparisons presented in the previous section, the performance of Raspberry Pi with the latest SBC like PandaBoard [6] and BeagleBone [7] was also compared in this paper as depicted in Figure 15. From this figure, despite some lack of performances Raspberry Pi is way cheaper than any of them.

After analysis, all the above performances, we came up with the following general conclusion about the advantages of Raspberry Pi:

- First, say that Raspberry Pi is a little autonomous PC that keeps running on the Linux working framework and can be customized as required.
- It has a huge working memory (numerous other sensor hubs don't have).
- It has substantial memory to save the information.
- It takes a shot at a processor which underpins a substantial arrangement of instructions.
- It works at rates from 700 MHz to 1000 MHz.
- It has bolster for USB 2.0 which permits its development with a substantial number of peripherals.
- Depending on the necessities it is conceivable to extend the Raspberry Pi with Wi-Fi and Bluetooth connectors (power and range can be altered by altering the connector).
- Expansion and correspondence with network gadgets over a LAN connector are conceivable.
- It is conceivable to shape an expandable framework with different electronic segments (sensors and electronic circuits) utilizing computerized information sources and yields, I2C or SPI conventions [19, 20] (most the today's gadgets utilize one of these strategies for correspondence).

For these reasons, it has been decided to go with Raspberry Pi as the SBC for sensor nodes in this work.

6. CONCLUSION

Right now, in the market, there are numerous industrially accessible sensor hub platforms. The work in this paper proposes Raspberry Pi - a shabby, adaptable, completely adjustable and programmable little PC board which has the capacities to be utilized as sensor hubs. The Raspberry Pi conveys the upsides of a PC to the space of SN, what makes it the ideal stage for interfacing with a wide assortment of outside peripherals. A similar examination of its key components and performances with a portion of the ebb and flow existing remote sensor hubs as displayed in this paper have demonstrated that in spite of few hindrances, the Raspberry Pi stays as a cheap PC with its exceptionally fruitful use in SN area and different scope of research applications.

The examination performed in this paper has demonstrated that other than the power utilization issue, Raspberry Pi is ultra-shoddy yet-serviceable PC board. With support for an expansive number of information and yield peripherals and system correspondence, it makes the ideal stage for interfacing with a wide range of gadgets and utilizing as a part of the extensive variety of uses. By coupling it with Wi-Fi or Ethernet links it can convey remotely, which makes Raspberry Pi exceptionally reasonable for the development of sensor hubs. Besides, Raspberry Pi can be utilized as handling hub as a part of SNs as sensor hub as well as a controller as appeared in this paper. Moreover, information handling and basic leadership on a Pi can be founded on A.I.

Likewise, the Linux working framework utilization gives extra favorable circumstances of utilizing Raspberry Pi as a sensor hubs. Programming in high state dialects, for example, C, C++, Python, or Java, arrangement execution is entirely basic and it is empowered to an extensive number of clients, restricted to microcontroller programming which as a rule relies on upon the advancement pack.

By introducing the Web Service on the server and giving access to the Internet on the sensor units, Raspberry Pi got to be finished and perfect framework (equipment and programming) for building the sensor hubs of the programmed entryway security framework in the real-world implementation of this research work as depicted in figure 16; with both near user interface and remote authenticate user interface detection features.

7. ACKNOWLEDGEMENTS:

The author would like to thank the dean and the other faculty members and office staffs of the Engineering Faculty of the Sakarya University of Turkey where this experiment was conducted and implemented as a part of the author's M.Sc. thesis research project.

8. REFERENCES

[1] Farha Ali, "A Middleware to Connect Software Applications with Sensor Web", The International Journal of Technology, Knowledge and Society, Volume 6, Issue 5, pp.27-36.

[2] Warneke, B. A. and Pister, K. S. J., "MEMS for Distributed Wireless Sensor Networks", Proceedings of the 9th IEEE International Conference on Electronics, Circuits and Systems, Volume 1, Dubrovnik, Croatia, 15-18 September 2002, pp. 291-294.

[3] M. Richardson and S. Wallace, Getting started with Raspberry Pi, O'Reilly, USA, 2013.

[4] M. Maurya, S. R. N. Shukla, "Current Wireless Sensor Nodes (Motes): Performance metrics and Constraints", International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE), Volume 2, Issue 1, January 2013.

[5] Memsic – Powerful Sensing Solutions, Available: http://www.memsic.com/wireless-sensor-networks, Access Date: 15.12.2013.

[6] "PandaBoard", PandaBoard's Homepage, Last time accessed: April 2015. [Online]. Available: http://www.pandaboard.org/ Access Date: 19.05.2015.

[7] G Coley, BeagleBone Black System Reference Manual. http://wiki3.icbanq.com/data/ICBShop/board/Biggle%20Bone%20Black%20DataSheet.pdf Access Date: 19.05.2015.

[8] M. Schmidt, Raspberry Pi – A Quick Start Guide, The Pragmatic Bookshelf, 2012.

[9] Hole, K.J., Dyrnes, E., and Thorsheim, P., "Securing Wi-Fi Networks," IEEE Computer, vol. 38, pages 28-34, July 2005.

[10] Fluhrer, S., Mantin, I., and Shamir, A., "Weaknesses in the key scheduling algorithm of RC4," In Eighth Annual Workshop on Selected Areas in Cryptography, August 2001.

[11] Raspberry-Pi-B-Plus-V1.2-Schematics http://www.raspberrypi.org/documentation/hardware/raspberrypi/schematics/Raspberry-Pi-B-Plus-V1.2-Schematics.pdf Access Date: 19.05.2015.

[12] MFRC522 Program Data Sheet. http://www.alldatasheet.com/datasheet-pdf/pdf/346106/NXP/MFRC522.html Access Date: 19.05.2015.

[13] E Upton, G Halfacree. Raspberry Pi user guide. books.google.com. 3rdEdition. 2014.

[14] Raspberry Pi Getting Started Guide, RS Components, Vsn 1.0, 2012.

[15] IBEX, Electronic Product Design Specialists. Raspberry Pi Resources. http://www.raspberry-projects.com/pi/pi-operating-systems/win32diskimager, Access Date: 15.05.2015.

[16] J. L. Hill, System Architecture for Wireless Sensor Networks, PhD Thesis, University of California, Berkley, 2003.

[17] B. Horan, Practical Raspberry Pi, Apres, USA, 2013.

[18] Luiz Felipe Perrone, David M. Nicol, Network modeling and simulation: a scalable simulator for TinyOS applications, Proceedings of the 34th conference on Winter simulation: exploring new frontiers, December 08-11, 2002, San Diego, California.

[19] UM10204 I2C-bus specification and user manual, Rev.5, October 2012.

[20] Serial Peripheral Interface, MC68HC11A8 Technical Data, Motorola, http://wwwee.eng.hawaii.edu/~tep/EE491E/Notes/HC11A8/HC11A8_SPI.pdf Access Date: 5.12.2013.