

Control Unit for a Home Automation System

Supporting ZigBee and Wi-Fi

Master Thesis

By:

Bondan Suwandi

Matriculation Number: 10035922

*A dissertation submitted in partial fulfillment of the requirements of South Westphalia
University of Applied Sciences for the award of Master of Science in Systems Engineering &
Engineering Management*

Supervisor: Prof. Dr. Ulf Witkowski

Second examiner: Prof. Dr. Werner Krybus

South Westphalia University of Applied Sciences

25 February 2014, Soest

This page is left blank intentionally

Abstract

Home automation system is a system which accommodates automation, monitor, and control of appliances in a house. These functionalities provide flexibility for the user to control and monitor home appliance wirelessly from the wireless local area network to the internet network. Example for this system is to monitor door lock status and gives possibility to lock it from the office through internet network. Other implementation of this system is to increase the amenities of elderly and disable people by helping them to control and monitor their home devices. This reason becomes main objective in this master thesis project. To actualize this purpose, a home automation system is designed. This system is expected to help impaired people who have difficulties to walk to check and control their house appliances wirelessly without moving from their place.

This thesis project focuses on developing a main control unit for home automation system. The control unit is based on Freescale microcontroller MK60FN, connected with ZigBee node sensor network and wireless local network through Ethernet and Wi-Fi modem. The system is developed to control 3 devices (simulated with LEDs) from a website and LCD touch panel. It is also designed to display the condition of the house (for example temperature and devices status) through website and LCD module. A web server is made in order to support the website functionalities. Moreover, ZigBee is used as a solution for low energy wireless communication system to handle the sensors network within the house.

At the end of this master thesis report, there are explanations on functionality test and result which had been made to test the robustness of this system. Some project experiences and suggestions are also given as a part of future work development.

Attestation

"No portion of the work presented in this dissertation has been submitted in support of another award or qualification either at this Institution or elsewhere."

Signature

Acknowledgement

I would like to acknowledge and express my high gratitude to the following persons who had help and made this Thesis possible to be completed:

Professor Ulf Witkowski, my thesis supervisor, for the opportunity to work under his supervision, the chance to work in his lab, and the possibilities to learn more and more.

Mr. Teerapat Chinapirom and Mr. Reza Zandian for their help and valuable advices during this master thesis project. I really appreciate their helps during the “dark era” of my project. So many things I have learned from their knowledge and experiences.

Mr. Engelbert Vahle, my lab. Administrator, for helping me in purchasing the components and giving permission to use equipments in the lab.

Mr. Ralf Stemmer and Mr. Sebastian Gebauer, my project partners, for their support and corporation to develop this thesis project for 4 months.

DAAD (Deutscher Akademischer Austausch Dienst), for the wonderful chances to stay, learn, and study in Germany. I have learned a lot about Germany, not only in the University but also in the social life.

My parents, family, and fiancée, for they endless support. Without it, I will not able to finish this master thesis on time.

Contents

Abstract	III
Attestation	IV
1. Introduction.....	3
1.1. Background	3
1.2. Motivation.....	4
1.3. Objective and Scopes	4
1.4. Project Management	5
1.5. Document Structure	5
2. Literature Review and Study.....	7
2.1. Home Automation System in General	7
2.2. Fundamental Development of Control Unit for Home Automation System	9
2.2.1. Freescale K60 Tower development board.....	10
2.2.2. ZigBee Module	16
2.2.3. LCD and Touch Screen Module	17
2.2.4. UART 232 to Serial Communication.....	18
2.2.5. Ethernet IC Driver.....	18
2.2.6. Current Sensor	19
3. Hardware Design	21
3.1. Schematic design	21
3.1.1. Power Supply Management Circuit	21
3.1.2. Freescale PK60FN1M0VLQ12 Circuit.....	22
3.1.3. Ethernet Communication Circuit	24
3.1.4. ZigBee Communication Circuit	25
3.1.5. Serial Console Circuit (UART232 to USB).....	26
3.1.6. Current Sensor Circuit (optional).....	27
3.1.7. GPIO and Additional Function Design	27
3.2. PCB design.....	28
3.2.1. First Layer	28
3.2.2. Second Layer	30
4. Software Design.....	31

4.1.	MCU Program.....	31
4.1.1.	TWRK60f120M Library (BSP/PSP) Setup	31
4.1.2.	Web server/Main Task.....	35
4.1.3.	IO Task.....	42
4.2.	Webpage Program.....	48
4.3.	LCD Touch Screen Program.....	49
5.	Assembly, Test, and Result.....	54
5.1.	Assembling process	54
5.2.	Functionality Test	56
5.2.1.	MCU tasks, digital-analog IO, and Console test.....	56
5.2.2.	LCD UART communication test	58
5.2.3.	ZigBee UART communication test.....	60
5.2.4.	Ethernet network test	63
5.2.5.	Web server and overall system test.....	66
5.3.	Problem Faced and Lesson Learned	70
6.	Conclusion and Future Works.....	73
	Appendices.....	74
	Schematic of Home Automation Control Unit	75
	List of Figures.....	78
	List of Tables	80
	Acronym and Abbreviation.....	81
	References.....	83

1. Introduction

Control unit for a home automation system is a system that designed to control several features and functions in a modern house. It provides a home base network and utilizes information from home appliances. It connects the entire household appliances one to each other and gives possibility to attach an internet access which also means possibility for the system to be accessed from outside internet world [1]. This chapter will define the background, motivation, objective, scope, and project management of the master thesis project. This chapter will also give a brief introduction about the thesis report structure.

1.1. Background

Nowadays, providing facility and amenity to the elderly and disable people are become a special and interesting challenge. The system development itself gives possibility to expand and grow for other purposes. One of the challenges is to develop a home automation system which gives facility to control and monitor their houses. This system will help them to collect, control, and monitor the status of their home devices wirelessly without giving direct contact to each appliance. In example, a system provides control system to turn on/off lamps, close/open doors, than gives status feedback through local or internet web server. By applying this home automation system to their houses, autonomy and satisfaction of elderly and disable people are expected to be increased.



Figure 1 - Home automation system in general [2]

Due to the functionality and interestedness of this topic, there are several home automation system methods which had been developed and have the same basic function. Those technologies are: Home

automation with an internet application [3], IP based Home automation system [4], home automation system based on Bluetooth technology [5], and a home automation system based on GSM Wireless network [6]. However, there are still many new technologies and methods can be applied to improve the home automation system performance.

One of the methods that will be used in this master thesis is home automation using ZigBee node communication and low cost - low energy microcontroller system. This system introduces a very low cost and low power consumption technology which can support two way wireless communication standards. Moreover, ZigBee technology also introduces a network layer (NWK) which is supports star, tree, and mesh network layer topologies [7].

1.2. Motivation

Basically, the motivation of developing this control unit for home automation system is to provide amenities for the elder, impaired, or disables people. The main purpose is to facilitate them with an easy-to-use system for home appliances control and monitor. Moreover, chances to learn the technology which is used and works with hardware and software development are the biggest consideration to choose this topic. Within this project, several knowledge can be reached encompass the hardware and software design, ZigBee low cost-low energy technology, web server development, problem solving, and so on.

1.3. Objective and Scopes

The aim of this thesis is to develop a control unit for home automation system. The system is integrated with a web server that can be used for home appliance automation system. This home control unit is also connected to the ZigBee host with one or several ZigBee nodes. The ZigBee node is equipped with many sensors and provides home status data. This integrated system allows user to interface sensors and actuators in the house, then control and monitor home appliances trough a web server.

There are several scopes and goals included in this master project which are:

1. Literal study of home automation technology, web server, and ZigBee communication.
2. Selection of modules which are used in the designed system.
3. Hardware design based on Freescale K60 development tower kit with ZigBee, Ethernet communication, and additional support functions.
4. Additional support functions are: digital and analog I/O interfaces, Serial LCD interface, current sensors and relay interface.

5. Software design for home automation system including the web server, I/O management, LCD serial, and ZigBee serial communication algorithms.
6. PCB fabrication, board assembly, test, demonstration, and thesis project documentation.

1.4. Project Management

One of the key factors of success in a master thesis project is time management. This time management has big a role in determining the timeliness of master thesis project. This master thesis is officially started from October 24th, 2013 and planned to be finished at February 25th, 2014. Due to the limitation of time, some literature study is done before the official date started.

Figure 2 below shows the thesis project time management base on Gantt-chart form.

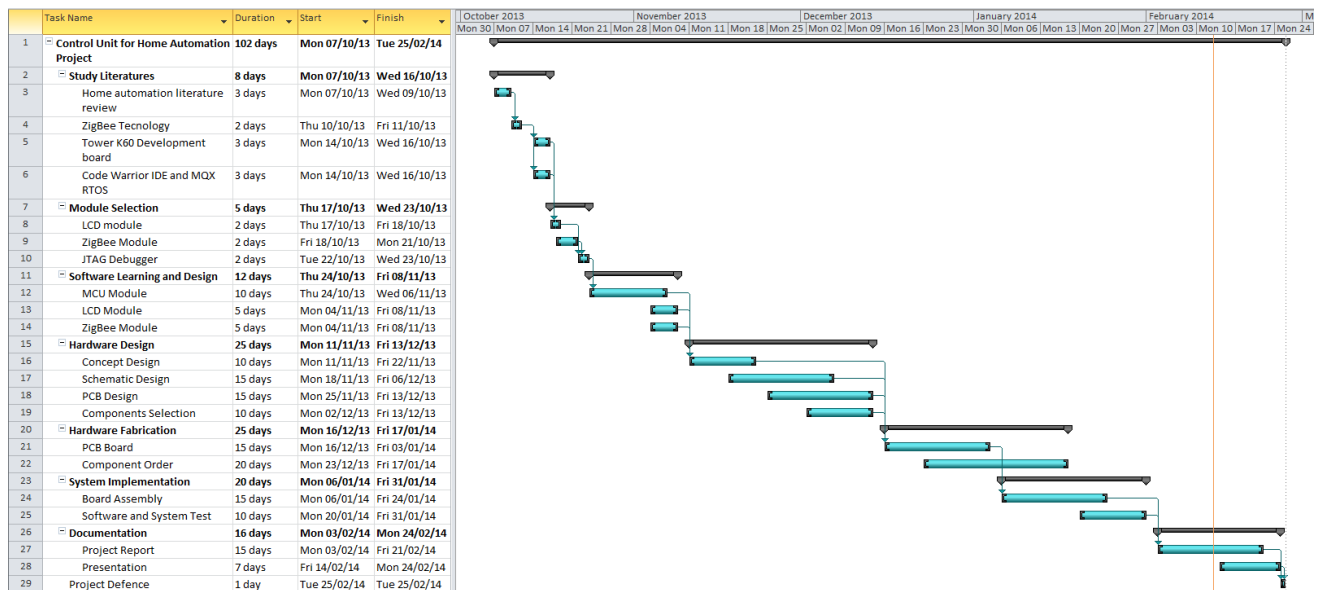


Figure 2 - Master thesis project Gantt-chart

1.5. Document Structure

This thesis is written and organized in a way that the reader can easily understand what had been done in this master thesis project. This thesis structure report provides project explanation, designed hardware and software, test and result, problem faced, lesson learned, and suggestion in order to provide full information for further study.

There are 6 chapters in this master thesis report which will be detail explained below:

1. Chapter 1: Introduction. This chapter describes the background, motivation, objective, scopes, project management, and document structure of this master project.

2. Chapter 2: Literature Review and Study. This chapter describes all the literatures that have been learned in this master project including the general development of home automation and devices which are used in this master project.
3. Chapter 3: Hardware Design. In this chapter, there are explanations about how the board is design, both in schematic and PCB layout specification.
4. Chapter 4: Software Design. This chapter describes how the software algorithm of the home automation system, including the web server and communication with other devices.
5. Chapter 5: Assembly, Test, and Result. There are three main parts on this chapter which includes the assembly process, functionality test, and problem faced and lesson learned.
6. Chapter 6: Conclusion and Future works. This chapter provides the conclusion of the whole master thesis project, including suggestion for further development for this home automation system.

2. Literature Review and Study

There are two main literature studies that had been made during this master thesis project. First literature study is to review home automation technologies and methods which had been developed, and the second literature study is more related to the home automation system that is developed in this master thesis project.

2.1. Home Automation System in General

No one knows when home automation development system started and the idea of controlling home appliances for elderly and disable people is not a span-new idea. Graffmans, with his book [8], introduce “gerontechnology” which comes from the combination between gerontology and technology science. He introduces two approaches for realizing intelligent house for elderly and disables people. First introduction is using special architecture solution to adapt the needs of those people, and second is using particular innovative technology to support their independent life. On this case, home automation system is a part of innovative technology that can help elderly and disable people to manage their daily routines.

Things which make home automation system difference one to each other are the methods that people use to develop this system. These methods include communication, device, protocol, and so on. According to [1], there are 7 basic type of home networking technology. Those technologies are:

1. Direct Cable: directly connecting 2 computers or devices through serial, parallel, or USB port. Inexpensive technology.
2. Ethernet: using hub system to connect 2 or more computers or devices. Network cards needed in each device. Driver installation and wiring required. Possibility in hardware conflict.
3. AC Network: using power line communication which already installed within home. Difficult to setup, slow data communication, and possibility to get interference from another device. Expensive.
4. Phone Line: shares data with phone line frequency. Special card, driver, and phone are required on each node.
5. Radio Free Network: using radio frequency to transmit and receive data. Up to 300 meters range data transmit through wall and doors. Require radio network card and possible to have some interference.
6. Spread Spectrum Wireless: Fast rate data transfer but limited in range.
7. Bluetooth: Wireless standard for cell phone and PDA devices. Possibility to make a node network.

Due to the wide-range and growth of technology, nowadays it is possible for developers to combine methods and generate more complex and robust home automation system. Latest technology also has been introduced and applied in the development of home automation system.

The most common development method that is normally used in home automation system is by integrating an internet communication network and create IP based home automatic system [3, 4]. This method is cheaper than the one which is developed based on GSM Wireless network [6]. Another method that had been introduced is a home automation system based on Bluetooth technology [5]. Bluetooth technology provides possibility for the node devices to communicate with the host within 10m of coverage (expandable to 100m by increasing the transmitted power).

Nowadays, people start to search for a new better solution in developing home automation system. The criteria itself mostly related to the low cost, low energy, long range, flexibility, and user-friendly devices. Referring to [9], more standardized communication protocols are used in the development of home automation system. Those communication protocols are “LON-bus, “BACnet”, EIB/KNX” and “ZigBee”. Between those technologies, Bluetooth ZigBee seems to be popularly used in the home automation system. This happened due to some advantages provided by ZigBee such as [9]:

1. Low wireless data processing rate sensors and control network. This network protocol is designed to consume small amount of power. ZigBee node expected to be last up to 4 months of operation.
2. Compared to Bluetooth which used for peer to peer network and Wi-Fi which is dedicated for internet data communication, ZigBee is much more intended for the wireless sensor network that already standardized.
3. ZigBee module provides wide coverage that can be used up to 100m for indoor use and up to 1500m in open field.
4. ZigBee has mesh networking feature and bridging feature. This bridging functionality enable virtually unlimited long distance range as long as nodes in the network are connected each others.

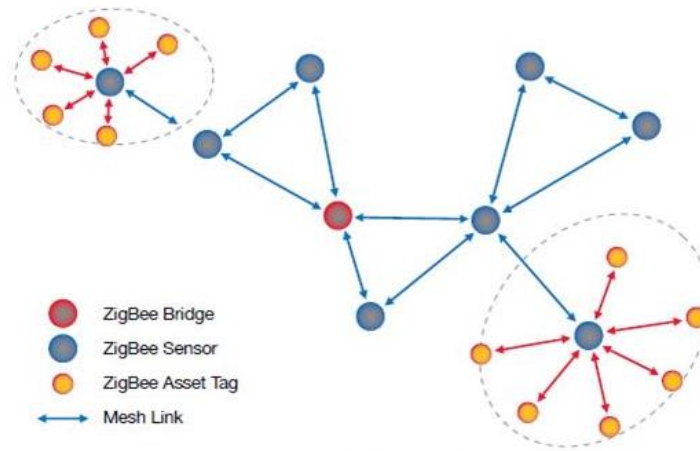


Figure 3 - ZigBee mesh and bridging network [10]

2.2. Fundamental Development of Control Unit for Home Automation System

The control unit for home automation system development is divided into 2 development parts which are software and hardware. In the software development part, software is developed to pool, control, and monitor the home appliance status through serial console, ZigBee host module, and LCD touch module. Then, this information are distributed to the web server and displayed in the website as visual-information for the user. This information are also sent and displayed to the LCD module. On the hardware part, the MCU for control unit system is based on microcontroller that supports Ethernet communication. This interface provides Ethernet connection to the local area network/internet through wireless network system (Wi-Fi). It also provides digital and analog I/O interfaces, USB console, and UARTs communication interfaces to communicate with other support system/component such as ZigBee and LCD module.

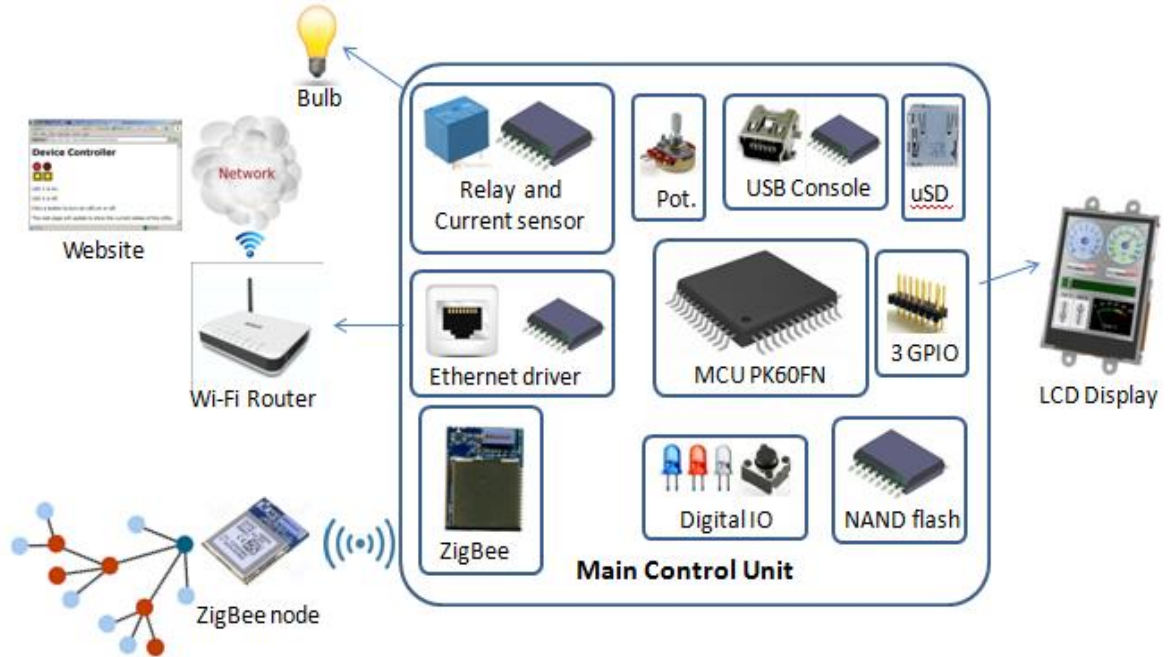


Figure 4 - Designed home automation system

2.2.1. Freescale K60 Tower development board

Control unit system development is based on K60 development board from Freescale. The explanation through Freescale K60 tower development board itself will be divided into 2 parts which are hardware and software.

2.2.1.1. Hardware

The microcontroller that is used in this master thesis is Freescale PK60FN1M0VLQ12 which supports Ethernet communication line. This MCU is the same microcontroller which is used in Freescale TWR-K60F120M tower development board. This development board itself includes TWR-SER extension board which provides Ethernet IC communication driver.

Freescale PK60FN1M0VLQ12 microcontroller comes from 120 MHz ARM® Cortex™-M4 core family and has some key features such as [11, 12]:

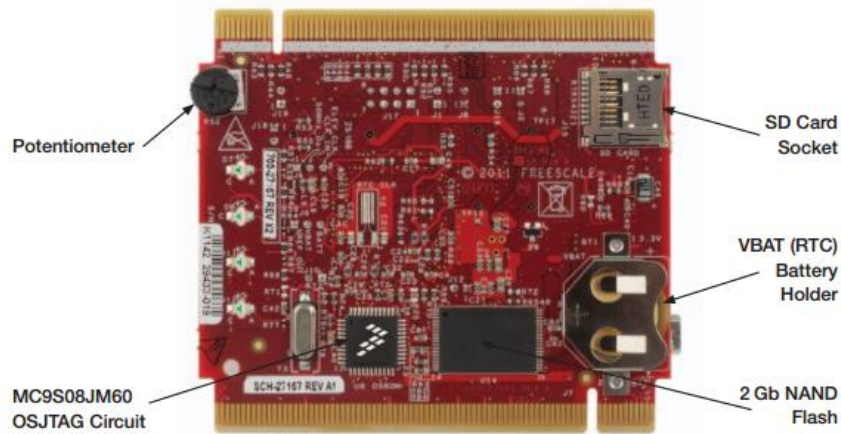
- Floating point unit, tamper detection, encryption.
- 1MB Flash, NAND Flash controller, Serial programmer controller (EzPort), FlexBus external bus interface.
- Ethernet, USB OTG, 3 x SPI, 6 x UART, 2 x CAN, 2 x I2C, 2 x I2S, and SDHC Modules.
- 144 LQFP package.

Freescale TWR-K60F120M development board has several features that support microcontroller functionalities such as [11]:

- MC9S08JM60 open source JTAG (OSJTAG) circuit
- Micron MT29F2G16ABAEAWP 2 GB NAND flash
- Four user-controlled status LEDs
- Four capacitive touch pads and two mechanical push buttons
- Three-axis digital accelerometer (MMA8451Q)
- General-purpose TWRPI socket (Tower plug-in module)
- TWRPI-TOUCH-STR socket (touch-sensing Tower plug-in)



(a)



(b)

Figure 5 - Front side (a) and back side (b) of TWR-K60F120M development board [11]

2.2.1.2. Software

The IDE that is used to develop the software for this microcontroller is CodeWarrior 10.3 Beta version. This IDE is based on Eclipse 4.2 Juno [13] which contains a base workspace and extendable

plug-in system and provides flexibility in customizing the environment [14]. Commonly, Eclipse is written in Java programming, but it also supports others language such as Ada, C, C++, COBOL, Phyton, and many else.

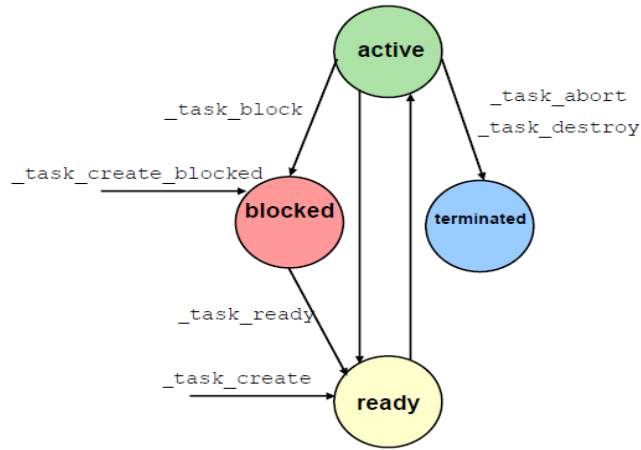
Freescale introduced an operating system called The MQX™ Real-Time Operating System (MQX™ RTOS). According to MQX RTOS User-Guide [15], “MQX RTOS has been designed for uni-processor, multi-processor and distributed-processor embedded real-time systems”. Several features of this MQX RTOS are:

- MQX RTOS includes run-time library functions that can be used in real-time multi-tasking applications.
- MQX can be combined with Board Support Package (BSP) and Project Support Package (PSP) which are used to handle the supporting part (driver, tasks, stacks, GPIO initialization, etc) of the development/custom board.
- The MQX software run-time libraries include: RTCS network stack, Shell interface library, USB (Host and Device) drivers, MS-DOS File system library (MFS) [16].
- MQX is scalable size, easy to use, and component-oriented architecture.
- MQX supports multi-processor applications, this provide development flexibility in networking, file management, and data communication.
- Used together with CodeWarrior, MQX provides a simple and flexible way to develop an application in the top of this RTOS.

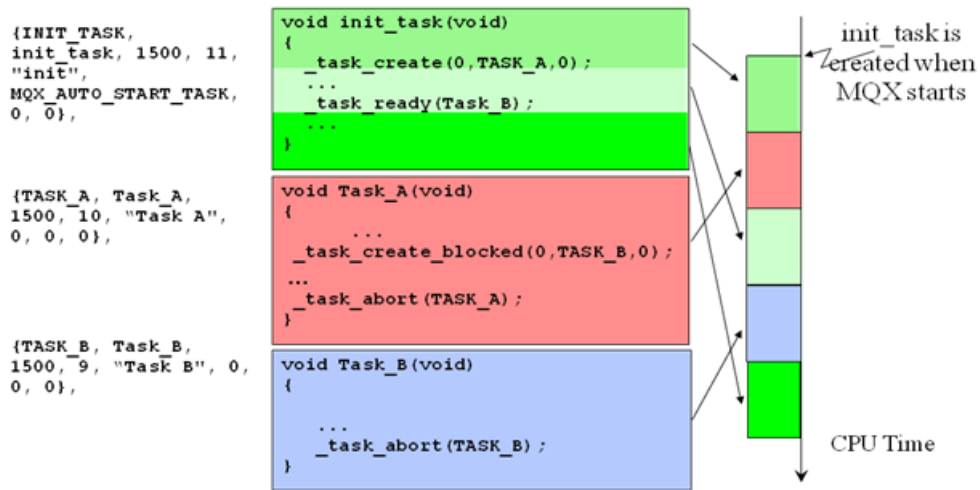
MQX tasks system

MQX RTOS provides a system called tasks system. This gives possibility for the system to have several tasks which work interchangeably depending on the task priority. There are 4 status / logic state for the task, these state are [17]:

- Active : the task is ready and active due to the highest priority ready task
- Ready : the task is ready but not running because it is not the highest priority
- Blocked : the task is blocked and not ready, waiting for the condition become true
- Terminate : the task is aborted or destroyed



(a)



(b)

Figure 6 - Four stack logic status (a) and usage illustration of tasks (b) [17]

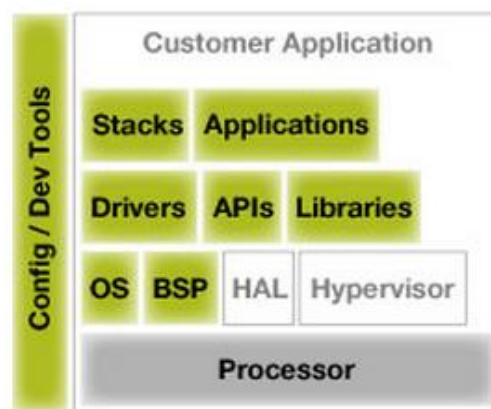
Figure 6(b) above is used to illustrate how tasks work in MQX RTOS. The illustration itself explained as follow:

1. Three tasks are initialized. These tasks are: INIT task (with priority 11), task A (with priority 10), and task B (with priority 9).
2. INIT task runs first. This happens because it was initialized to run when the MCU is started (MQX_AUTO_START_TASK).

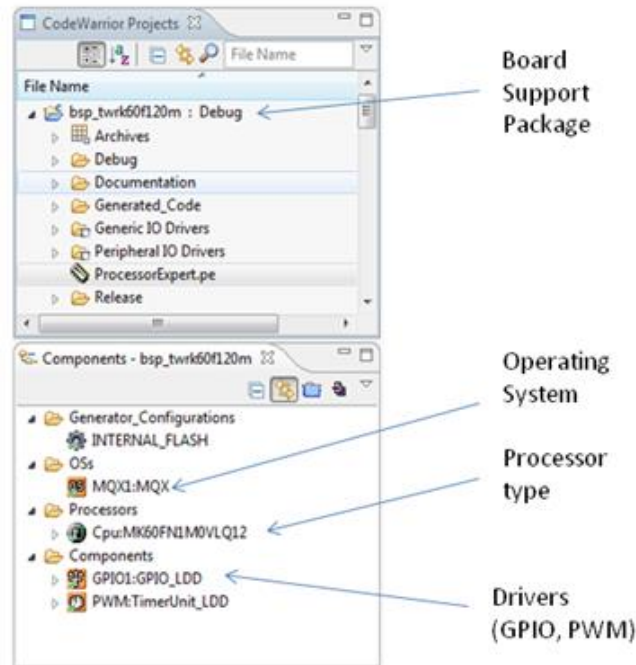
3. When task A is created by INIT task, MCU will hold the code execution in INIT task and start the task A. It happens because task A has higher priority than task B.
4. Inside the task A, task B is generated with blocked status. This create task B in blocked condition.
5. When task A is finish, the execution code is going back to INIT task.
6. Inside INIT task, task B status is changed to ready status. This makes the execution jump to task B, because task B has higher priority than INIT task.
7. Task B is executed until this task is finish.
8. When the task B finish, MCU will go back to INIT task and continue to execute the code inside this task.

Board Support Package (BSP)

One of the run-time library functions in MQX RTOS is Board Support Package (BSP) which controls the basic functionalities of the processor. MQX RTOS provides tasks and BSP provide drivers, APIs, and some function libraries. Both this functionalities can be used by the programmer to develop their basic application code. There are some configuration files to set the parameter of BSP. Some of these files are *user_config.h* and *init_GPIO.h*. Inside BSP, there is also ProcessorExpert system which has functions to create, configure, optimize, migrate, and deliver software/drive component, and then generate source codes [18]. Figure 7 explain how level of development software is developed by Freescale.



(a)



(b)

Figure 7 - Freescale software development level [19] (a) and Illustration of BSP / driver configuration (b)

MQX Real-Time TCP/IP Communication Suit (RTCS)

The Real-Time TCP/IP Communication Suit (RTCS) is a part of MQX RTOS run-time library [16]. This communication suit provides IP networking for MQX software solution. Some key features of RTCS are:

- Specifically designed to adding TCP/IP connectivity to embedded systems.
- Provides full features set of networking stack which configurable to fit the small memory in embedded devices.
- Support lower-layer protocols such as Ethernet (IEEE 802.3) and PPP.
- Integrated with MQX RTOS device drivers for Ethernet and other access layer.
- Provide network status and diagnostics.

Basically, RTCS library is a stack-driver which has a function to drive the Ethernet functionalities. Once RTCS initialization is done, RTCS can run and control a web server automatically, depend on the setting that had been made in the initialization process.

2.2.2. ZigBee Module

ZigBee module that is used in this control unit development board is DZ-SB-SA from DiZiC. This module has ZigBee RF module with standard IEEE 802.15.4 and provide quick add wireless networking capabilities. This feature means that this product is ready-to-use, simple to operate, and available in wide configuration range [20]. Some specifications of this module are:

- 2.4GHz CSS Transceiver.
- 32 bit ARM® Cortex™-M3 processor (STM32W108).
- 128 kB Flash, 8 kB RAM memory.
- AES128 encryption accelerator.
- ADC, UART, timer, and GPIO.
- Rx sensitivity (-100 dBm), Tx power lever (+7dBm).
- Low power consumption, less than 400nA at low deep sleep.
- Certified for following standard: Radio EN 300 328:v1.7.1, EMC EN 301 489-17:V2.1.1, safety EN 60950-1:2005(Ed.2.0).

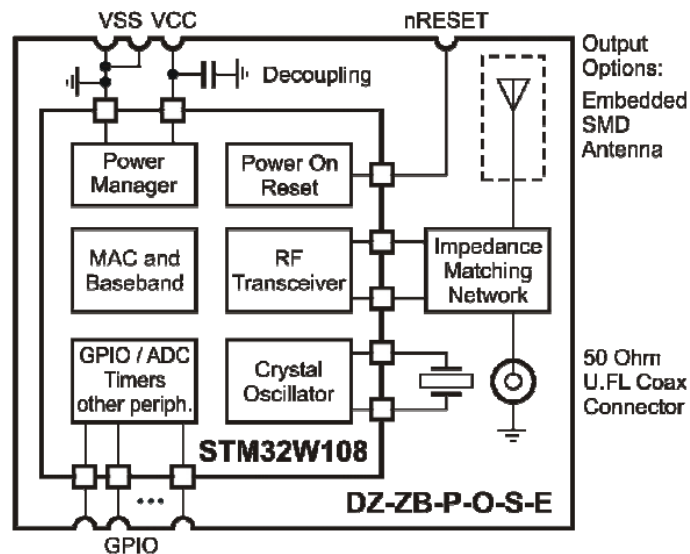


Figure 8 - Simplified ZigBee module block diagram [21]

In this project, ZigBee module communicates with microcontroller through UART communication protocol. Some commands are developed as a way to communicates between these two devices.

2.2.3. LCD and Touch Screen Module

uLCD-32PTU module is a display and touch screen module developed by 4D-Systems. This module is an independent module which controlled by PICASSO processor to control the functionalities of this LCD. This processor also provides UART communication feature to communicate with other device such as MCU. In this project, LCD module communicates with MCU by using UART communication. Almost similar with ZigBee-MCU communication method, some UART commands are prepared to help LCD to communicate with MCU. Other features that provide by this module are [22]:

- Low-cost 3.2” LCD-TFT display graphics with 65K true color, 240x320 VGA resolutions.
- Integrated 4-wire resistive touch panel.
- Easy 5 pin interface with any serial host devices.
- 4D-labs PICASSO processor with 14kB of SRAM.
- 2 x UART, I2C, 8 x 16 bit timers, 13 x GPIO, and onboard micro SD FAT16 memory card adaptor.



Figure 9 - uLCD-32PTU LCD and touch module [22]

To develop the functionalities of this LCD display, an IDE from 4D systems called “Workshop 4 IDE” is used. Workshop 4 IDE has 4 multiple development environments for the programmer. Those environments are explained below [23]:

- **Designer:** Environment that provides natural form of programming to program uLCD-32PTU.
- **ViSi:** Visual programming that enables the programmer to drag-and-drop objects in the workspace and generates the code to the programming language. This allows the programmer to visualize how the template display will look likes.

- **ViSi-Genie:** An advance programming environment which is not requires any coding. The code generate automatically. Programming is made by drag-and-drop object and set the event to drive them.
- **Serial:** In this environment, LCD will be a serial slave module. This allows the user to control the display for any microcontroller device as a master through serial port communication.

2.2.4. UART 232 to Serial Communication

One of the UART functions in this thesis project is assigned as a serial console. This serial console is supported with UART 232 to serial USB converter. Serial console is needed to help to monitor the events and processes which are happened inside the MCU. UART comes from Universal Asynchronous Receiver Transmitter which literally means a communication protocol that translates data from a parallel to serial stream across the communication link [24]. It calls asynchronous because each devices on the link are generating their own independent frequencies, and new events arrive asynchronously to the local clock.

To support this functionality, a serial UART to USB IC from FTDI (series FT232RL) is chosen. FT232RL has several advanced features such as [25]:

- Single chip USB, asynchronous serial data interface.
- Ready to use, no specific USB firmware required. Entire USB protocol handled on the chip.
- Data transfer rate from 300 Baud to 3 Mbaud (RS422, RS485, and RS232) at TTL levels.
- No external clock requires. Optional clock output selection for interface flexibility to external MCU or FPGA.
- UART interface support 7 or 8 data bits, 1 or 2 stop bits and odd/even/mark/space/no parity.

2.2.5. Ethernet IC Driver

Freescall PK60FN1M0VLQ12 MCU supports Ethernet data connection. This MCU provides Ethernet controller with media Independent Interface (MII/Reduced-MII) to external PHY with standard IEEE 1588 hardware capability [12]. In order to support this capability, an Ethernet IC driver from Micrel (series KSZ8041MLL) is chosen. According to KSZ8041MLL datasheet [26], this Ethernet IC driver support IEEE 802.3 MII interface which also known as Management Data Input / Output (MDIO) interface. This interface allows MCU to monitor and control the state of Ethernet IC drive. MCU is also allowed to read the PHY status and configure the PHY setting through this interface. MII management interface in KSZ8041MLL consist of:

- Physical connection that incorporates MDIO data line with the clock line (MDC)

- Specific protocol that operates through physical connection. This protocol allows MCU to communicate with more than one Ethernet IC devices. Each IC devices is assigned a PHY address by the PHYAD[2:0] strapping pins.
- Internal addressable which set of thirteen 16-bit MDIO register. The first 7 bit register are required and defined by the IEEE 802.3u specification. Else are provided for expanded function.

The MII data interface itself has some following key characteristics, such as:

- Support 10Mbps and 100Mbps data rates.
- Use 25 MHz reference clock.
- Consist of two groups of signal which are transmission and reception.
- Provide independent 4-bit data path for transmit and receive.

Table 1 below shows MII signal definition for KSZ8041MLL.

Table 1 - KSZ8041MLL MII signal definition [26]

MII Signal Name	Direction (with respect to PHY, KSZ8041TL/FTL/MLL signal)	Direction (with respect to MAC)	Description
TXC	Output	Input	Transmit Clock (2.5 MHz for 10Mbps; 25 MHz for 100Mbps)
TXEN	Input	Output	Transmit Enable
TXD[3:0]	Input	Output	Transmit Data [3:0]
RXC	Output	Input	Receive Clock (2.5 MHz for 10Mbps; 25 MHz for 100Mbps)
RXDV	Output	Input	Receive Data Valid
RXD[3:0]	Output	Input	Receive Data [3:0]
RXER	Output	Input, or (not required)	Receive Error
CRS	Output	Input	Carrier Sense
COL	Output	Input	Collision Detection

2.2.6. Current Sensor

Current sensor in this home automation development board is used to measure the current consumption of a device. The device can be varied such as bulb, stove, actuator, and more. The sensor itself is using Allegro ACS716KLATR-6BB-T which has specification as follow [27]:

- Precision linear Hall sensor integrated circuit with a copper conduction.
- High level immunity to current conductor dV/dt and stray electric fields means low ripple output.

- Possibility to define over-current input fault threshold through VOC pin. When there is over-current in the input, the open drain over-current fault pin will set to logic low state.
- 1Mohm primary conductor resistance for low power loss.
- +- 6A maximum current measurement.
- 277 VAC continuous voltage ratings (per UL standard 1577)

Current sensor and relay module are included to the home automation control unit design. However, there are no implementations applied on this module.

3. Hardware Design

The hardware design is done by using Eagle PCB design software from CadSoft. Eagle PCB design software provides both schematic and PCB design, including the common part libraries. It also provides flexibility to develop custom component libraries. Eagle PCB design software and its component libraries are available from the official website with some limitation features for the free version [28].

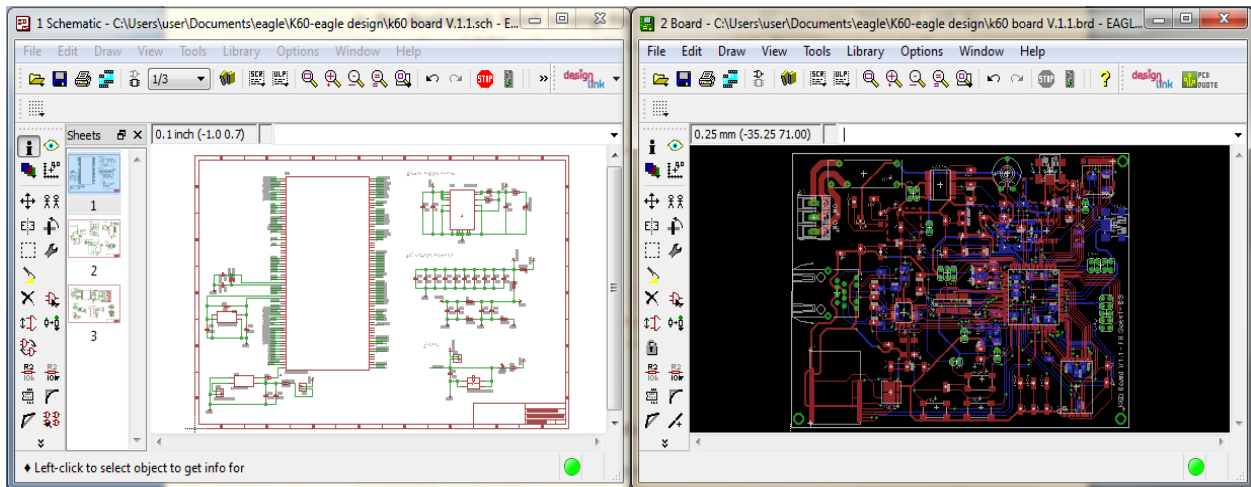


Figure 10 - schematic and PCB design with Eagle from CadSoft

3.1. Schematic design

There are 7 main parts that will be explained briefly in this schematic design sub-chapter. These parts include the minimum design requirements for each component to run the system. Each of this system is also supporting another system, one to each other.

3.1.1. Power Supply Management Circuit

Power supply management system is one of the most important parts in developing the control system for home automation. This system is “the pulse of life” for the entire system. Inspired from AMIRE robot development board, power supply management system in this custom board is using the same step down DC converter TPS62111 with fix output voltage 3.3VDC and up to 1.5A output current [30]. The voltage input itself comes from 5VDC USB source.

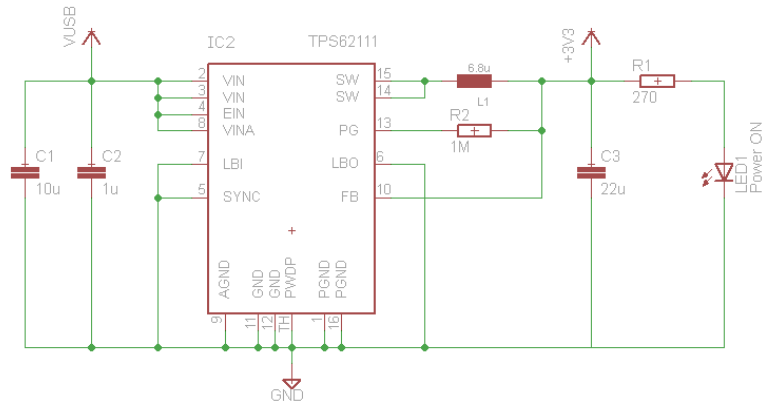


Figure 11 - Power supply management circuit schematic

3.1.2. Freescale PK60FN1M0VLQ12 Circuit

PK60FN1M0VLQ12 is the main MCU in the control unit home automation system development. This MCU controls all of the function in the system that includes communication to the ZigBee host module, Ethernet IC Driver, UART console, ADC, GPIO, and others. There are 3 external oscillators supporting this microcontroller which are 12MHz oscillator, 32 MHz RTC oscillator, and 50MHz oscillator. These oscillator assignments are base on default BSP external oscillator setup in MQX 3.8.1 operating system for MCU PK60FN1M0VLQ12. Other circuit that supporting this MCU is a reset circuit with pull-up circuit characteristic. Pull-up circuit is designed in order to deal with the active low reset characteristic of this MCU. Figure 12 below shows the functional assignment of MCU PK60FN with other functionalities in the control unit for home automation board.

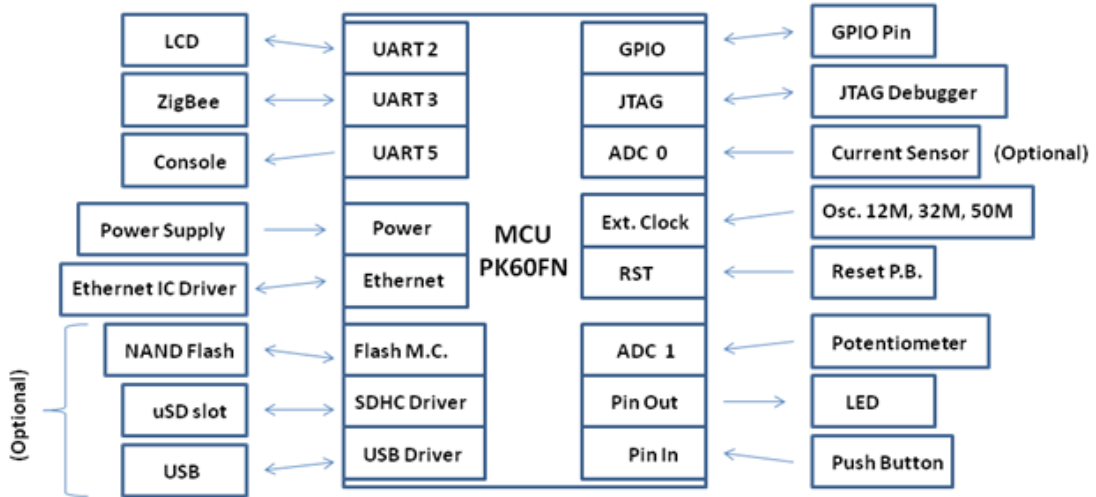


Figure 12 - Functional assignment of MCU PK60FN

For software developing purpose, some detail and functionality of pin assignment in MCU PK60FN are given in Table 2 below.

Table 2 - Functionality pin assignment for MCU PK60FN1M0VLQ12

Function	Components	MCU Pin Assignment	Remarks
UART 2 (LCD comm.)	UART 2 TX	PTD 3	
	UART 2 RX	PTD 2	
	UART 2 CTS	PTD 1	
	UART 2 RTS	PTD 0	
UART 3 (ZigBee comm.)	UART 3 TX	PTB 11	
	UART 3 RX	PTB 10	
UART 5 (Terminal Console)	UART 5 TX	PTE 8	
	UART 5 RX	PTE 9	
	UART 5 CTS	PTE 10	
	UART 5 RTS	PTE 11	
Pin Out (Digital Out)	LED 3	PTA 11	
	LED 4	PTA 28	
	LED 5	PTA 29	
	OUT Relay	PTE 6	
Pin In	PB 1	PTE 26	

(Digital In)	PB 2	PTA 19	
Potentiometer	ADC	ADC3_DM0	
Current Sensor	ADC	ADC2_DM0	
Additional GPIO	-----	-----	Additional GPIO
<i>SPI communication</i>	SPI 2 PCS 1	PTD 15	
	SPI 2 SIN	PTD 14	
	SPI 2 SOUT	PTD 13	
	SPI 2 SCK	PTD 12	
	SPI 2 PCS 0	PTD 0	
<i>CAN communication</i>	CAN 0 TX	PTB 18	
	CAN 0 RX	PTB 19	
<i>I2C communication</i>	I2C 0 SDA	PTB 3	
	I2C 0 SCL	PTB 2	
<i>ADC</i>	ADC 0	ADC0_SE16	
	ADC 1	ADC1_SE16	
<i>DAC</i>	DAC 0	AC0_OUT	
	DAC 1	DAC1_OUT	

Other functionalities, such as Ethernet MII communication, NAND Flash, uSD slot, USB, Reset, JTAG Debugger, External Oscillator, and Power supply are assigned directly to the dedicated pins in MCU. These pins are assigned with standard assignment as shown in the PK60FN documents [12, 30].

3.1.3. Ethernet Communication Circuit

In Ethernet circuit design, there are some pin setup that need to be applied in the IC hardware. This setup related to the communication type is used in this Ethernet IC driver. Based on KSZ8041MLL datasheet [26], this Ethernet IC driver only support MII communication mode. To set this MII configuration, PIN 27 (CONFIG2), PIN 41 (CONFIG1), and PIN 40 (CONFIG0) should be 0 or not connected (NC) to any other sources. Other hardware characteristic that needs also to be mention is that the value of the external oscillator should be 25 MHz for MII configuration. Some capacitors and ferrite beads are also needed in this circuit to reduce the high frequencies signal noises. Direction of MII Ethernet data connection itself is based on Freescale PK60FN datasheets [12, 31] and Ethernet IC driver datasheet [26].

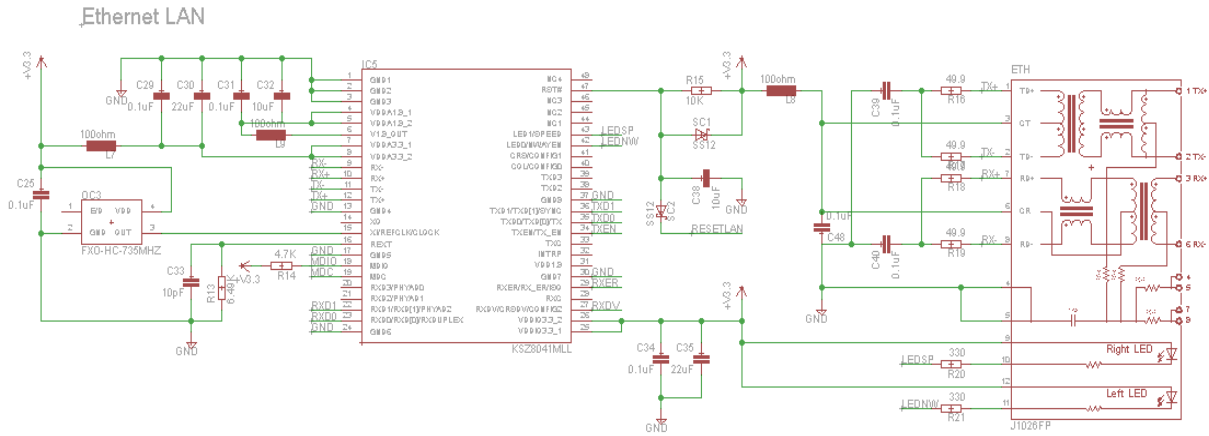


Figure 13 - Ethernet IC driver schematic design

3.1.4. ZigBee Communication Circuit

ZigBee module and MCU PK60FN are connected by using UART communication in channel 3 (UART 3). This communication requires 2 lines of communication which are transmit and receive. The pin assignments detail can be seen in table 2 above (chapter 3.1.2). One consideration that this communication is using UART crosses data lines. In example, pin TX from MCU should be connected to pin RX in DZ-ZB-SA module. It also applies in vice versa.

Other connection that needs to be provided for this module is JTAG debugger connection. This connection allows user to debug and upload ZigBee node software into the controller inside the ZigBee node module (Microcontroller STM32W108). Figure 14 below shows the schematic diagram of ZigBee module in control unit home automation system.

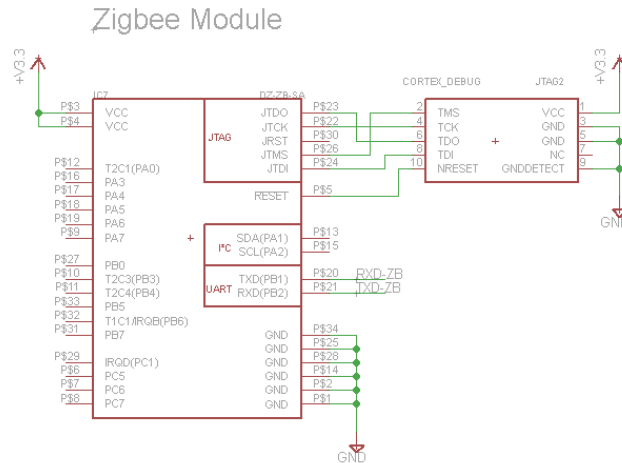


Figure 14 - ZigBee module schematic diagram

3.1.5. Serial Console Circuit (UART232 to USB)

This section explains UART 232 serial to USB converter circuit diagram. As mentioned before, FT232RL IC is used as UART to USB IC converter. There are 4 UART inputs connected to this IC which are Transmit data (TXD), Receive data (RXD), Request to send (RTS), and Clear to send (CTS). Both TXD and RXD are used to receive and transmit data, meanwhile RTS and CTS are used as control line communication. This console UART is assigned as UART 5 in MCU PK60FN.

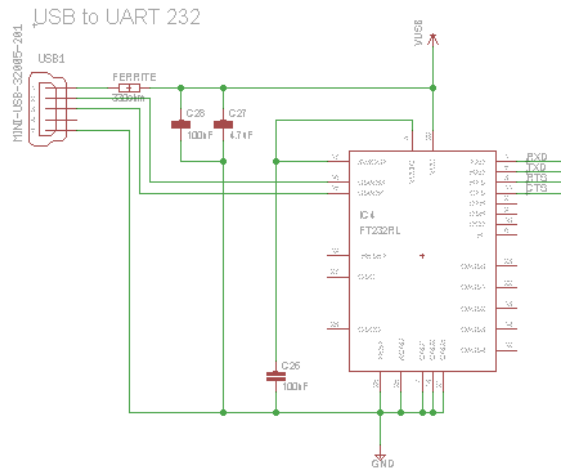


Figure 15 - Serial UART 232 to USB converter circuit diagram

The design of UART serial to USB converter IC, including the placement and the value of passive components, is based on FT232RL datasheet [25].

UART2 from MCU. Meanwhile, Micro SD module, USB module, NAND Flash module, CON2, and CON3 are additional modules. These modules are included to K60 control unit home automation board as part of further home automation development plan. The schematic for these modules are adopted and assigned based on TWR-K60F120M development board schematic design [30] and table 2.

Complete schematic design can be seen in the appendices part.

3.2. PCB design

The PCB is designed with dimension of 95 x 128 mm and has 2 layers for signal line and components placement. Two layers PCB is chosen to minimize over-stepping signal line, simplify the components placement, and more reasonable price compare to 4 layers PCB. Manual signal routes had been made in order to increase board space efficiency. There are some PCB design standards that need to be considered such as:

- Place the voltage source capacitors as close as possible to voltage inputs in MCU (less than 5mm away).
- Place the crystal and load capacitors as close as possible to the XTAL pins in MCU.
- Minimum signal line wide is not less than 0.2 mm.
- Bigger line width for VCC and GND signal.
- Avoid 90 degrees of turned lines.

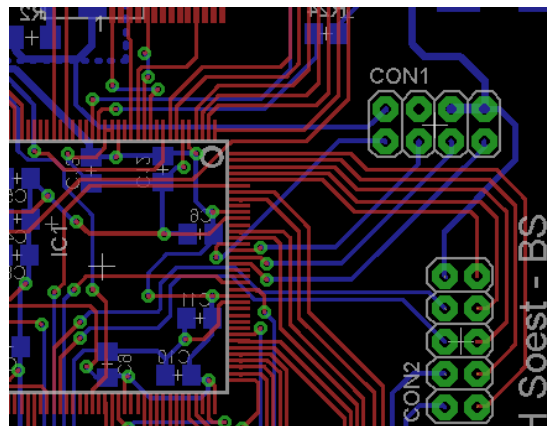


Figure 17 - Example of 45 degrees turned signal lines

3.2.1. First Layer

Most of modules in control unit board are located in the 1st PCB layer. These modules are located in top PCB side to provide easiness and flexibility on accessing it. These modules are:

1. Microcontroller MK60FN
2. UART to USB communication console
3. Ethernet IC driver circuit
4. ZigBee host module
5. Digital IO (LEDs and Push Button)
6. Analog input (ADC potentiometer)
7. Current sensor and relay circuit
8. uSD card module
9. NAND Flash module
10. General Purpose IO

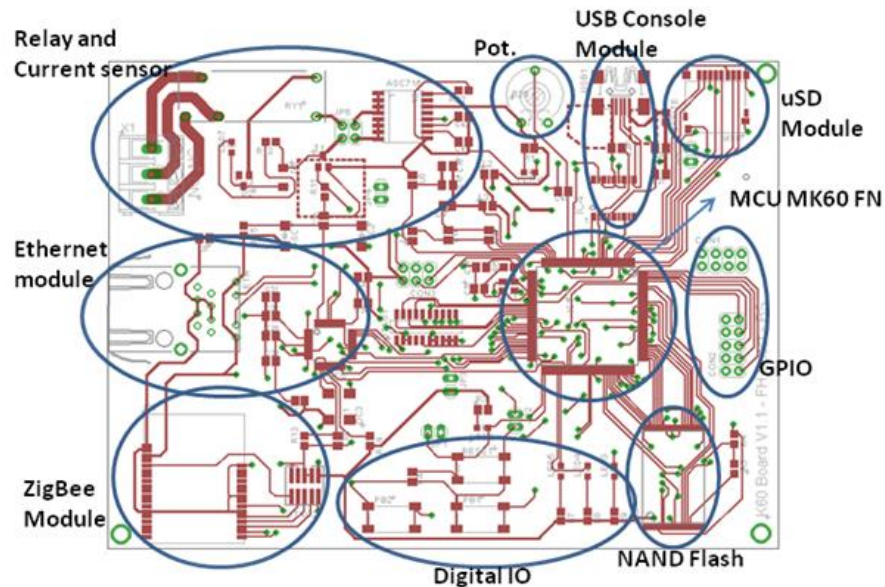


Figure 18 - Top layer modules placement

Figure above showed how the top layer modules are placed in the control unit board. There are several considerations that had been made during designing this board. For example, the MCU PK60FN is placed almost in the middle of the board because the MCU should be able to reach, control, and communicate with other modules. Other consideration is the ZigBee module should be placed in a quite open circuit area, especially in the antenna part. The antenna area should be free from any circuit line to avoid disturbances. Some signals are also designed as polygon area such as ground and VCC. These provide quite large areas for these signals to reduce noise and disturbance from voltage source.

3.2.2. Second Layer

Different with the 1st layer, in the 2nd layer there are only voltage regulator module and 2nd USB functionalities. The voltage regulator module is placed near to 1st and 2nd USB module to accommodate near 5VDC USB voltage source. Other components are 32 MHz and 50MHz MCU oscillator circuits. These circuits are placed in the bottom in order to make it closer to the MCU.

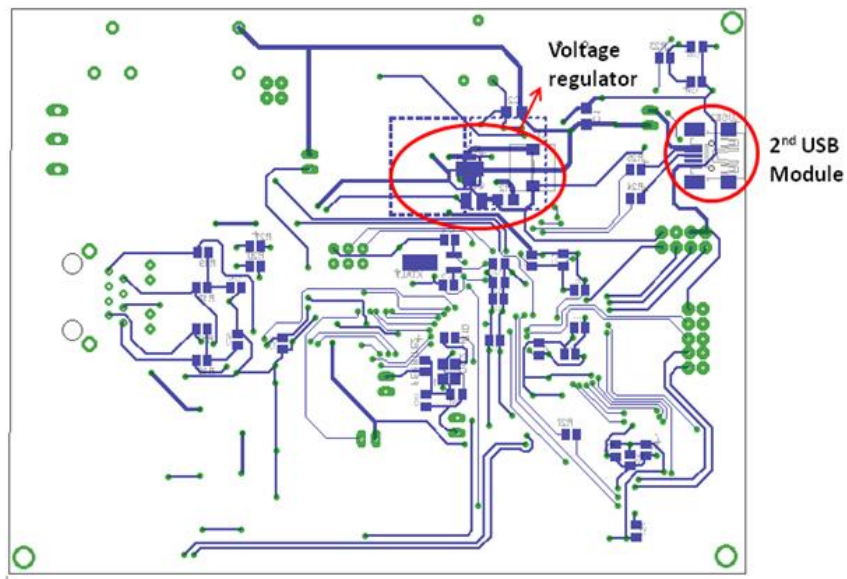


Figure 19 - bottom layer module placement

4. Software Design

This chapter explains how the home automation software flows and the system works. There are two main IDE which are used to develop the system. These IDE are CodeWarrior IDE and workshop 4 IDE. Worked together with MQX RTOS library, CodeWarrior IDE is used to develop the software for MCU PK60FN. On the other hand, Workshop 4 IDE is used to develop the software for touch LCD module.

4.1. MCU Program

Basically, MCU home automation software is made by modifying MQX demo security web server which is provided by the MQX RTOS software example [33]. There are 3 parts that will be explained in this main MCU program chapter. The first part is the setup of BSP and PSP in the MQX RTOS which focusing on configuration of the operating system, the assignment of the MCU pins, and MCU functions setup.

Other two parts are related to the programming the MCU tasks. As mentioned before, MQX RTOS supports multi-task programming which allow the microcontroller to do several different tasks. The tasks which are made in home automation system software are web server task and IO task.

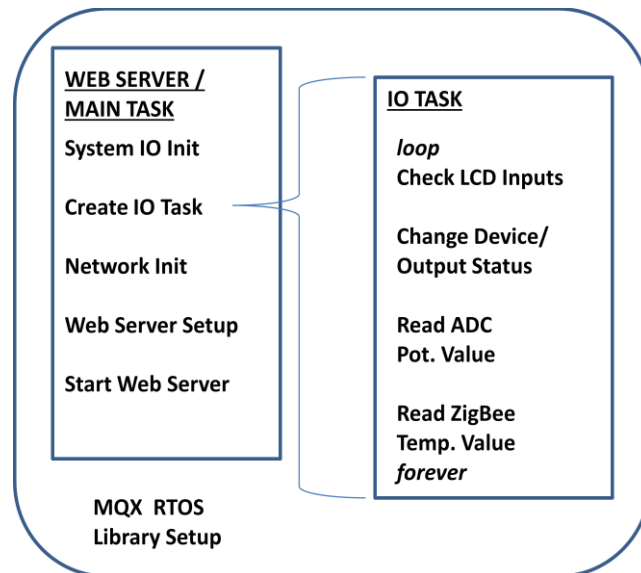


Figure 20 – Home automation system software tasks diagram

4.1.1. TWRK60f120M Library (BSP/PSP) Setup

The meaning of BSP comes from its abbreviation which is Board Support Package meanwhile PSP comes from Project Support Package. Basically, both BSP and PSP are libraries that support the board and simplify the programming process. Home automation custom board is based on

TWRK60f120M development board from Freescale. However, there are some changes and additional functionalities have been added to the custom board. That's why those BSP and PSP need to be re-setup.

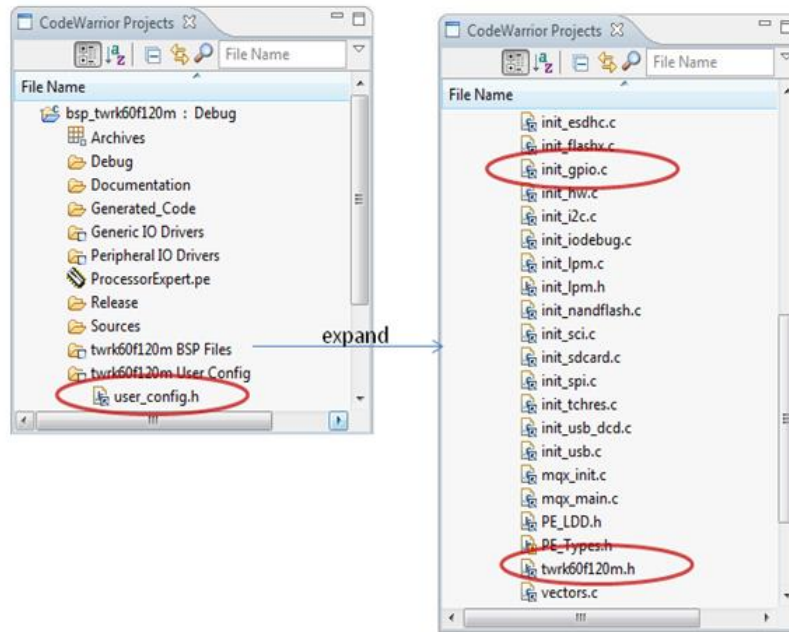


Figure 21 - BSP setup files

There are 3 files that need to be modified from BSP TWRK60f120M development board library, first is *user_config.h*, second is *init_gpio.c*, and the last one is *twrk60f120m.h*. *user_config.h* can be found under *twrk60f120m User Config* folder, meanwhile *init_gpio.c* and *twrk60f120m.h* can be found under *twrk60f120m BSP Files* folder.

The first step of this library initialization is to initialize the *user_config.h*. This file consists of library functionality setup for K60 control unit board. In this case, UART 2 and UART 3 need to be activated (set from 0 to 1). There are 2 type mode of UART communication provided by BSP, TTYx and ITTYx. Differences between these 2 UART modes are that TTYx mode is used in the polled UART mode meanwhile ITTYx mode is used in the interrupted UART mode. In MQX Operating system, UART 2 is represented as TTYC and ITTYC, and UART 3 is represented as TTYD and ITTYD. Both TTYx and ITTYx are set to enable to activate both mode of UART configuration. Different with UART 5, UART 5 does not need any configuration. This is because, by default, BSP for TWRK60f120M development board already assigned UART 5 as serial console port.

```
user_config.h
31
32 #ifndef __user_config_h__
33 #define __user_config_h__
34
35 /* mandatory CPU identification */
36 #define MQX_CPU                PSP_CPU_MK60DF120M
37
38
39 /* MGCT: <generated_code> */
40 #define BSPCFG_ENABLE_TTYA      1
41 #define BSPCFG_ENABLE_ITTYA    0
42 #define BSPCFG_ENABLE_TTYB     1
43 #define BSPCFG_ENABLE_ITTYB    0
44 #define BSPCFG_ENABLE_ITTYC    1
45 #define BSPCFG_ENABLE_ITTYC    1
46 #define BSPCFG_ENABLE_TTYD     1
47 #define BSPCFG_ENABLE_ITTYD    1
48 #define BSPCFG_ENABLE_TTYE     1
49 #define BSPCFG_ENABLE_ITTYE    0
50 #define BSPCFG_ENABLE_TTYF     1
51 #define BSPCFG_ENABLE_ITTYF    0
52 #define BSPCFG_ENABLE_I2C0     1
53 #define BSPCFG_ENABLE_I2C0     1
54 #define BSPCFG_ENABLE_I2C1     0
55 #define BSPCFG_ENABLE_I2C1     0
56 #define BSPCFG_ENABLE_SPI0     0
```

Figure 22 - *user_config.h*: UART 2 and UART 3 configuration

Second initialization is to modify the *init_gpio.c* file. This file consist all of the pin initializations and assignments for the MCU. By default from TWRK60f120M development board library, UART 3 is assigned to PORTC 16 for RX and PORTC 17 for TX [34]. However, in control unit custom board (table 2), UART 3 is set to PORTB 10 for RX and PORTB 11 for RX. In order to modify the pin, some initializations need to be changed in *init_gpio.c* file. Figure 23 below shows the initialization that need to be changed in *init_gpio.c* file.

In figure 23, case 3 represent the pin initialization of UART 3. PORTB_BASE_PTR is a variable defined by the BSP library to represent PORT B. Moreover, PCR [10] and PCR [11] represent the MCU port bit itself.

```
user_config.h | main.c | twrk60f120m.h | serial.h | init_gpio.c
142     case 3:
143         pctl = (PORT_MemMapPtr)PORTB_BASE_PTR;
144         if (flags & IO_PERIPHERAL_PIN_MUX_ENABLE)
145         {
146             /* PTC16 as RX function (Alt.3) + drive strength */
147             pctl->PCR[10] = 0 | PORT_PCR_MUX(3) | PORT_PCR_DSE_MASK;
148             /* PTC17 as TX function (Alt.3) + drive strength */
149             pctl->PCR[11] = 0 | PORT_PCR_MUX(3) | PORT_PCR_DSE_MASK;
150         }
151         if (flags & IO_PERIPHERAL_CLOCK_ENABLE)
152         {
153             /* start SGI clock */
154             sim->SCGC4 |= SIM_SCGC4_UART3_MASK;
155         }
156         if (flags & IO_PERIPHERAL_CLOCK_DISABLE)
157         {
158             /* stop SGI clock */
159             sim->SCGC4 &= (~ SIM_SCGC4_UART3_MASK);
160         }
161         break;
162
163     case 4:
164         pctl = (PORT_MemMapPtr) PORTC_BASE_PTR;
165         if (flags & IO_PERIPHERAL_PIN_MUX_ENABLE) {
166
167             /* PTC14 as RX function (Alt.3) + drive strength */
```

Figure 23 - *init_gpio.c*: UART3 pin assignments

The last step of this initialization is to modify *twrk60f120m.h* which can be found under *twrk60f120m BSP Files* folder. This file consists of all initialization modes which are used in the MCU. Some of these function modes are I2C mode, UART mode (baud rate, echo, and flow), ADC mode, and so on. For this home automation project, UART communication are set to perform software flow control (xon/xoff) and perform serial translation mode. Perform translation mode means the outgoing “\n” will be translated to CR\LF, incoming CR will be translated to “\n”, and incoming backspace will erase previous character. UART setup, options, and UART mode explanations can be found in *twrk60f120m.h* and *serial.h* files as shown in figure 24 and 25 below.

```

user_config.h | main.c | twrk60f120m.h | serial.h | init_gpio.c
949 #define BSP_DEFAULT_IO_CHANNEL_DEFINED
950 #endif
951
952 /*
953 ** MGCT: <option type="string" maxsize="1024" quoted="false" allowempty="false"/>
954 */
955 #ifndef BSP_DEFAULT_IO_OPEN_MODE
956 #define BSP_DEFAULT_IO_OPEN_MODE (pointer) (IO_SERIAL_XON_XOFF | IO_SERIAL_TRANSLATION)
957 #endif
958
959 /*
960 ** FLASHX flash memory pool minimum size
961 ** MGCT: <option type="string" maxsize="1024" quoted="false" allowempty="false"/>
962 */
963 #if BSPCFG_ENABLE_FLASHX
964 #ifndef BSPCFG_FLASHX_SIZE
965 #define BSPCFG_FLASHX_SIZE 0x4000
966 #endif
967 #else
968 #undef BSPCFG_FLASHX_SIZE
969 #define BSPCFG_FLASHX_SIZE 0x0
970 #endif
971
972 /** MGCT: </category> */
973 #endif /* _twrk60f120m_h_ */
974

```

Figure 24 - *twrk60f120m.h*: UART mode initialization

```

user_config.h | main.c | twrk60f120m.h | serial.h | init_gpio.c
44 /*
45 **          CONSTANT DEFINITIONS
46 */
47
48 /* Incoming and outgoing data not processed */
49 #define IO_SERIAL_RAW_IO (0)
50
51 /* Perform xon/xoff processing */
52 #define IO_SERIAL_XON_XOFF (0x01)
53
54 /*
55 ** Perform translation :
56 **   outgoing \n to CR\LF
57 **   incoming CR to \n
58 **   incoming backspace erases previous character
59 */
60 #define IO_SERIAL_TRANSLATION (0x02)
61
62 /* echo incoming characters */
63 #define IO_SERIAL_ECHO (0x04)
64
65 /* Perform hardware flow control processing */
66 #define IO_SERIAL_HW_FLOW_CONTROL (0x08)
67
68 /* */
69 #define IO_SERIAL_NON_BLOCKING (0x10)
70

```

Figure 25 - *serial.h*: options for UART mode initialization

4.1.2. Web server/Main Task

Web server/Main task has 5 main functionalities which are IO system initialization, generate the IO task, network initialization, web server initialization, and then run the web server using MQX RTOS-RTCS real-time library. Complete flowchart for this task can be seen in figure 26 below.

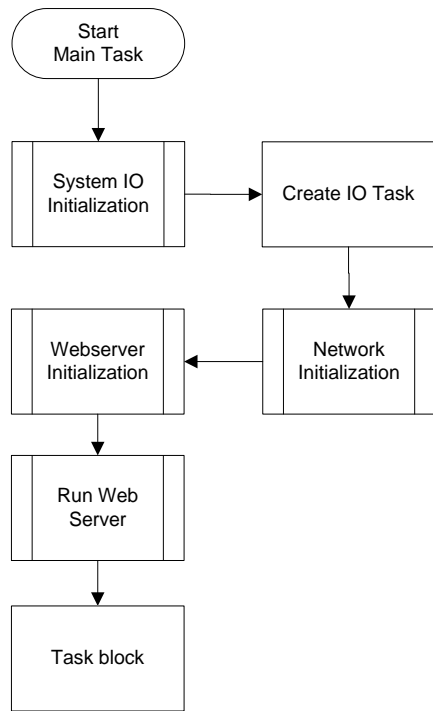


Figure 26 - Web server/main task flowchart

4.1.2.1. System IO Initialization

The first step for this main task is to initialize the IO system. This initialization includes GPIO initialization, UART initialization, ADC initialization, and default status initialization.

GPIO initialization

GPIO initialization is used to initial the digital output for the home automation system. Three LEDs will be used as indicators to simulate the digital output. A function has been provided by the BSP to initialize the GPIO called *lwgpio_init* (*LWGPIIO_STRUC_PTR*, *LWGPIIO_PIN_ID*, *LWGPIIO_DIR*, *LWGPIIO_VALUE*);. This function has a return value that will indicate FALSE when GPIO port is unable to open, and TRUE when GPIO is successfully open.

UART Initialization

As mentioned before, there are 3 UART use in this home automation project. UART 5 comes as a serial console by default from MQX BSP library. Meanwhile UART 2 and UART 3 are needed to be initialized. To initialize these UART, a function called *fopen* ("*ittyx:*", *BSP_DEFAULT_IO_OPEN_MODE*); is used. X represents the UART name that will be open, and *BSP_DEFAULT_IO_OPEN_MODE* represents the mode that will be used in the serial communication. This mode can be set up in *twrk60f120m.h* file as mentioned in chapter 4.1.1. *fopen*

function itself is defined and initialized by MQX PSP library. This function will return a NULL value when the UART port fails to open.

ADC Initialization

Similar with GPIO and UART, to activate the ADC function, ADC function also need to be open. The function to open this ADC function is *fopen (MY_ADC, (const char*)&adc_init);*. Here *MY_ADC* had defined to represent the “*adc1*” and *adc_init* indicates ADC mode which is set to 16bit resolution.

Default Status Initialization

The status initialization has a function to initialize default status of devices and variable values in the beginning of system started. Example of this initialization is to set all the digital output (LEDs) to OFF status and set all variable values to zero.

Flowchart for this initialization is shown in Figure 27 below.

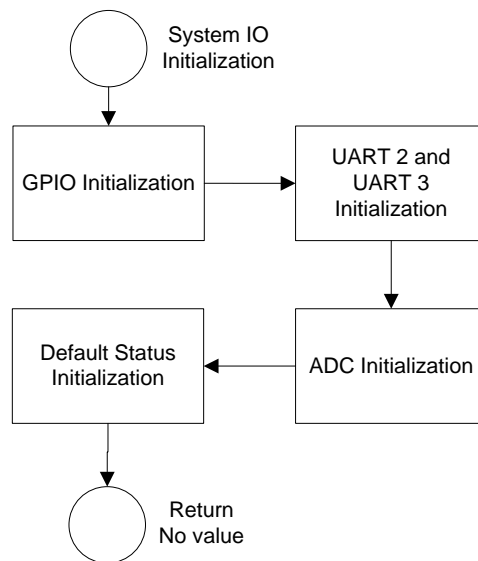


Figure 27 - GPIO initialization flowchart

4.1.2.2. Create IO Task

Create IO task is a function to start the second task which is IO task. This can be done by using a function called *_task_create (0, IO_TASK, 0);*. This function will return a NULL value if the task is failed to create.

```

95
96  _task_create(0, IO_TASK, 0); // start io task
97

```

Figure 28 - `_task_create` function

4.1.2.3. Network Initialization

The network initialization includes several processes to set the RTCS. This setup is required as a basic setup of the RTCS. These processes are [35]:

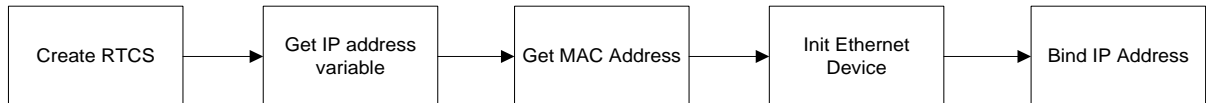


Figure 29 - RTCS setup processes [35]

The first setup step is to create the RTCS. This step can be done by using `RTCS_create()`; function which is provided by MQX RTCS library. When the RTCS is successfully created, next setup step is to get the network IP address. This RTCS is set as DHCP enable mode. RTCS will get the IP address automatically from the network and store it in `ip_data` structure.

After this process, next step is to initialize the Ethernet device. To initialize the Ethernet device, a function called `ipcfg_init_device (BSP_DEFAULT_ENET_DEVICE, enet_address)`; is used. This function tells the RTCS task to set the calculated MAC address to the device and tell which Ethernet device should be use. Once the device initialization is done and the MAC address is set, device can be bind to the IP address.

To bind the IP address to the Ethernet device, a function called `ipcfg_bind_dhcp_wait (BSP_DEFAULT_ENET_DEVICE, 1, &ip_data)`; is executed. This function requires the device number and `ip_data` structure as parameter. `ip_data` is a local structure that has parameters of the IP address configuration for the device. After the initialization, `ip_data` structure binds the Ethernet device with this IP address.

After the bind process complete, RTCS is ready to communicate through the Ethernet network. It is also possible to ping the IP address once it is connected to the line.

4.1.2.4. Web Server Initialization

Web server initialization includes TFS setup, HTTP server initialization, and CGI configuration. The flow of this initialization is shown in Figure 29 below.

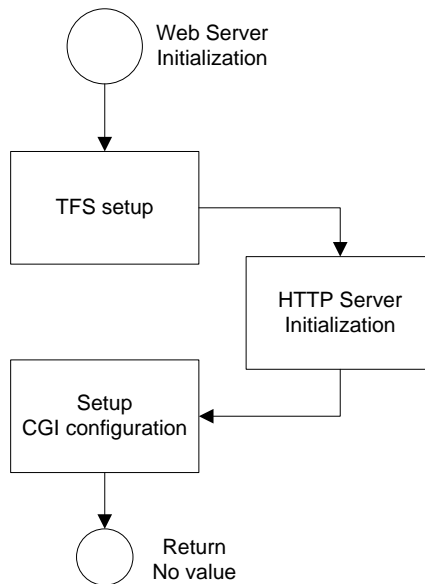


Figure 30 - Web server initialization flowchart

The first step in Web Server initialization is to initialize the Trivial File System (TFS). TFS data file holds the web page information as an array. To generate this file, MQX includes `mkfts.exe` application to convert web page files into a C source code file [34]. To install this TFS data, a function called `_io_tfs_install ("tfs:", tfs_data);` is used. This allows the RTCS to access the web page data which is stored in the “tfs:” partition.

When there is no error in the previous initialization, server needs to be initialized with the specified `root_dir` with the web page index page. On this project, the index page is called “\mqx.html”. Before the server runs, server need to be configured with the CGI information which contains the device status / information from the home automation system such as ADC value, devices (LEDs) value, and temperature value. This CGI data is provide by a function called `cgi_sec_data ();` that will be explained in the next sub chapter.

4.1.2.5. Run Web Server

Run Web Server is a function that runs the web server in the home automation system. This functionality is driven by MQX RTOS RTCS and use `cgi_sec_data ();` to collect devices data and to create a devices-status table. This table later will be used to update the CGI file. CGI data then is read by the java function in the webpage software, and then the webpage will display the data as home automation devices status / information. The data itself can be seen from the CGI file by accessing `<ip_address_webserver>/secdata.cgi`. Beside this functionality, `cgi_sec_data ();` also has a function to read the input data from the webpage. These inputs will be used to update the home

automation devices status. Figure 31 below shows how data flow from the device status variable until it is shown in the webpage.

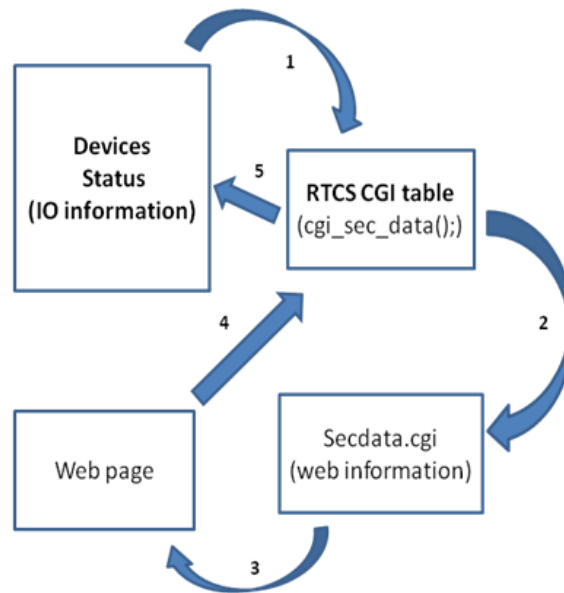


Figure 31 - RTCS server data flow

Main task data flow:

1. RTCS stack used `cgi_sec_data ();` function to check and update CGI table of home automation devices status (LEDs, Log, ADC, and Temp).
2. RTCS stack read the CGI table and update the information to CGI file (`secdata.cgi`).
3. Java code in the webpage takes the information in the CGI file and display it to the webpage
4. `cgi_sec_data ();` check whether there is any change from push button in the website.
5. When there is a change, `cgi_sec_data ();` changes the home automation device status.

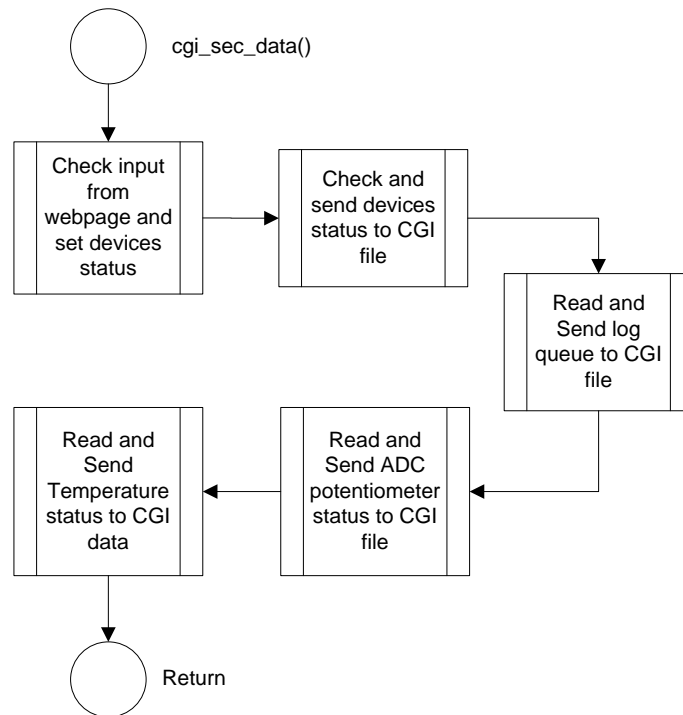


Figure 32 - Flowchart for *cgi_sec_data()*;

Figure 32 above shows the flowchart for *cgi_sec_data()*. The flowchart is explained below.

Check input from webpage and set devices status

This function is used to check the push button (radio button) input from the webpage. When a push button pressed in the webpage, a data will be sent with format “?Byte=XXXX”. This “XXXX” itself is the changes data status of the push button in the web page. Each X represents one push button. *cgi_sec_data()*; has the function to read this data called *POINTER= session->request.urldata;*. Once this sub-program read the data from webpage, *cgi_sec_data()*; will read the status of the webpage push button then update the home automation devices status.

Check and send device’s status to CGI file

Next process is to check and send the home automation device’s status. This function will check each device’s status (LEDs) and send this status to the CGI data table. The LED devices include LED 1, LED 2, and LED 3.

Read and send log queue, ADC potentiometer status, and Temperature to CGI file

Not so many different with the function above, these function also check the log queue from the home automation system. It also reads ADC value from the potentiometer and temperature measurement from ZigBee. These data then send to the CGI data table.

4.1.3. IO Task

Once IO task is created by the main task, the IO task will start its functionalities. This IO task has several functions which are to pool the UART 2 inputs from the LCD module (LCD Touch buttons), giving back responds to UART 2, measure ADC potentiometer value, and request temperature value from ZigBee through UART 3 communication. Data flow and Flowchart for this task can be seen in figures below.

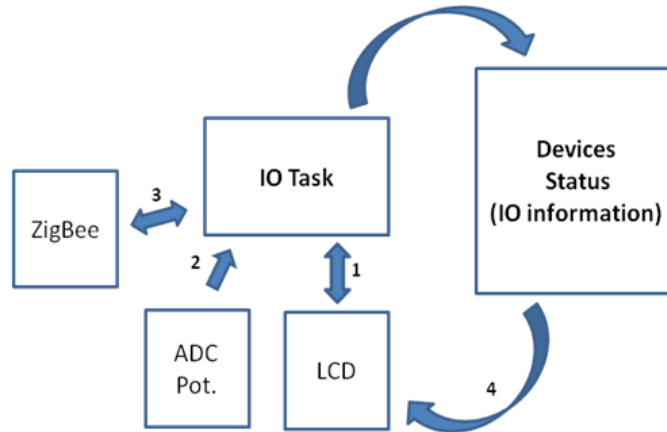


Figure 33 - IO task data flow

IO task data flow:

1. IO task function waits whether there are any incoming UART 2 data from LCD module. When there is a data, IO task will analyze the data and change the home automation device status. When the incoming data is requesting temperature value, IO task will send the temperature value to LCD module Through UART 2.
2. IO task function measures the value of potentiometer with ADC then update the ADC measurement value to the home automation device status.
3. IO task sends temperature request to the ZigBee host and wait until ZigBee host answer. Once it answers, IO task will use this data to update temperature value to the home automation device status.
4. Whenever there is a change in LED device's status, this function will also send a string code through UART 2 to LCD module. This will update the LED indicator in LCD display.

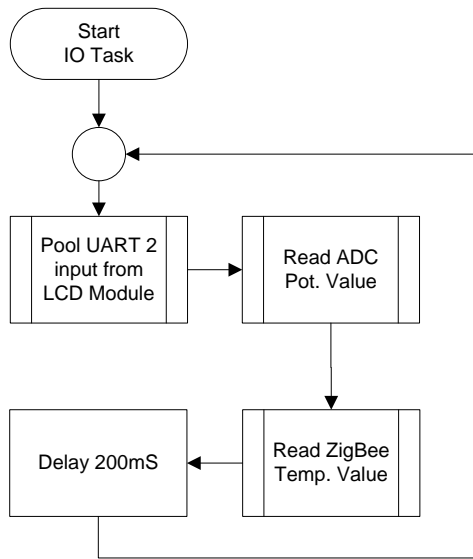


Figure 34 - IO task flowchart

There are three main sub-codes which construct the IO task. These sub-codes are Pool UART 2 input, read ADC potentiometer value, and read ZigBee temperature value. These sub-codes will be explained below.

Pool UART2 Input

The main task in this sub-code is to read the UART 2 input values from the LCD device. There are 4 inputs that indicate 4 touch buttons in the LCD. Three of this touch buttons represent the LED switches and one touch button indicates temperature request. Besides reading the UART inputs, MCU also provide data feedback to LCD module through the same UART communication line. Inputs from the LCD module and MCU data feedbacks are explained in table 3 below.

Table 3 - Received string values from LCD module

UART2 Input	Action	Feedback to LCD
"A0\r"	Set Device 1 OFF	"A0\r"
"A1\r"	Set Device 1 ON	"A1\r"
"B0\r"	Set Device 2 OFF	"B0\r"
"B1\r"	Set Device 2 ON	"B1\r"
"C0\r"	Set Device 3 OFF	"C0\r"
"C1\r"	Set Device 3 ON	"C1\r"
"RT\r"	Read temperature value	"(temperature value)"

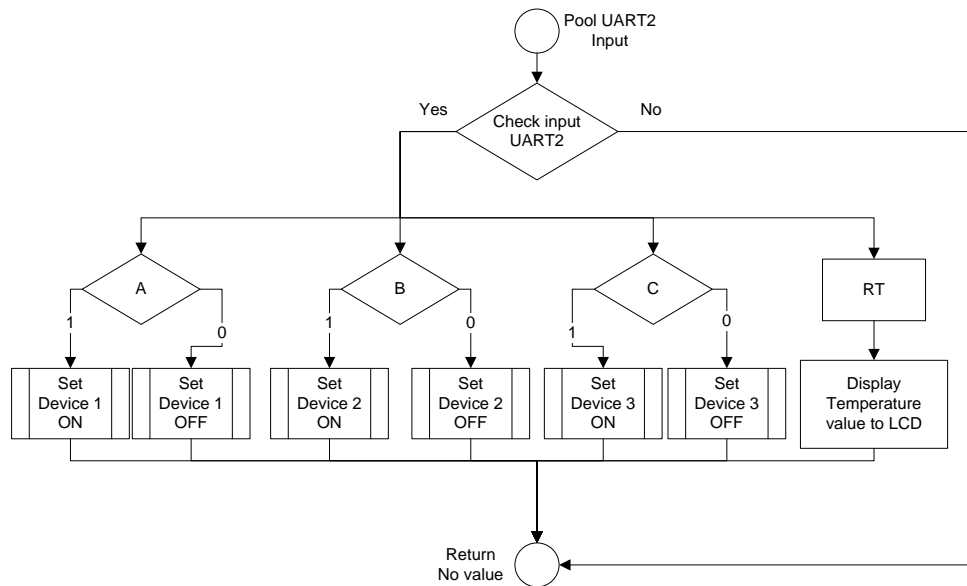


Figure 35 - Pool UART 2 Input flowchart

Explanation of flowchart above is given as an example. When LED1 touch button in LCD module is pressed ON, MCU will receive string data “A1\r”. MCU will read the first and second string data, and then run another function called *SEC_SetDevice1_ON()*. This function has functionality to update the status of device 1 as “ON” condition, add “device1 ON” information to the log table, print device 1 status to the console, and turn on LED number 1. When the status device 1 is set as “ON”, this function also send a string value “A1\r” to LCD module through UART 2. This process is to mention LCD module that LED1 indicator need to be switch ON. Figure 36 below shows the flowchart for *SEC_SetDevice1_ON()*; sub-code.

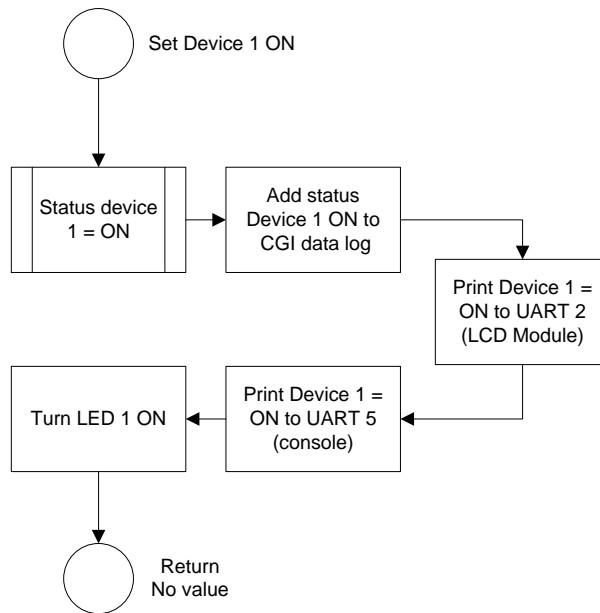


Figure 36 - Flowchart of *SEC_SetDevice1_ON()*;

Read ADC Pot. Value

This sub-software is called *ReadADC(_mqx_int channel)*;. This function has functionalities to read the ADC value from ADC3_DM0 pin and send the measurement value as a return. This pin is connected to the potentiometer in the home automation custom board. Figure 37 below shows the flowchart of the *ReadADC(_mqx_int channel)*;

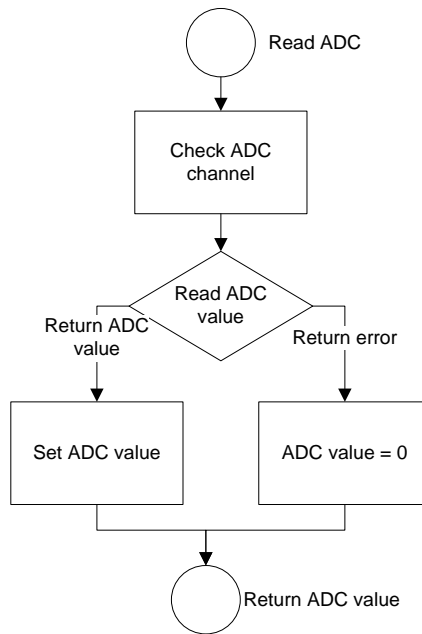


Figure 37 - Flow chart of read ADC function

Read ZigBee Temperature Value

This sub-code function has functionality to read temperature value from ZigBee host module. MCU communicates with the ZigBee host module through UART3 communication. To request the temperature value, a string command is sent to the ZigBee host module. List of the command are explained in table 4 below [36].

Table 4 - ZigBee command and the data response

Send Command to UART3	Response Data from UART3	Remarks
"hi\n"	"Hi\n"	Check communication function.
"secret.test.cmd\n"	"DATA: 0x00000001 0x00000000 0x00000001 0x1289ABEF\n"	Request temperature dummy data.
"ls.nodes\n"	"NODELIST: 0x0001 "SensNet-GPNode"\n" "NODELIST: 0x0002 "SensNet-GPNode"\n"	Request list of connected nodes.

	<p>...</p> <p>“NODELIST: 0x0010 “ ”\n”</p>	
<p>“send 00000001 00000001 0000001\n”</p>	<p>“DATA: 0x00000001 0x00000000 0x00000001 0xXXXXXXXX\n”</p>	<p>Request sensor value from: node 1, sensor ID 1 (temp), request command. XXXXXXXX is the temperature value.</p>
<p>“send FFFFFFFF 00000001 0000001\n”</p>	<p>“DATA: 0x00000001 0x00000000 0x00000001 0xXXXXXXXX\n” “DATA: 0x00000002 0x00000000 0x00000001 0xXXXXXXXX\n” “DATA: 0x00000010 0x00000000 0x00000001 0xXXXXXXXX\n”</p>	<p>Request sensor value from: all nodes (0x01 to 0x10), sensor ID 1 (temp), request command.</p>
	<p>“ERROR 00050009\n”</p>	<p>Error is received when there is no data received</p>

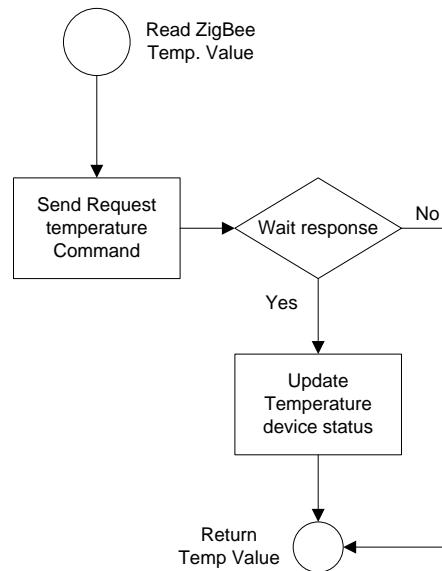


Figure 38 - Flowchart of read ZigBee temperature value

When a read ZigBee temperature value function is called, MCU will send a request command through UART 3 to ZigBee host module. On this case the command is “send FFFFFFFF 00000001 00000001\n”. MCU then wait until there are responses from the node. When the response comes, MCU will read the temperature value and update this value to the temperature status variable.

4.2. Webpage Program

The webpage of this home automation system is made by modifying example web from MQX demo security web server. This example is provided by the MQX RTOS software example [34]. A software editor called notepad++ is used to modify this webpage. There are several functionalities in this website such as input function and information display function.

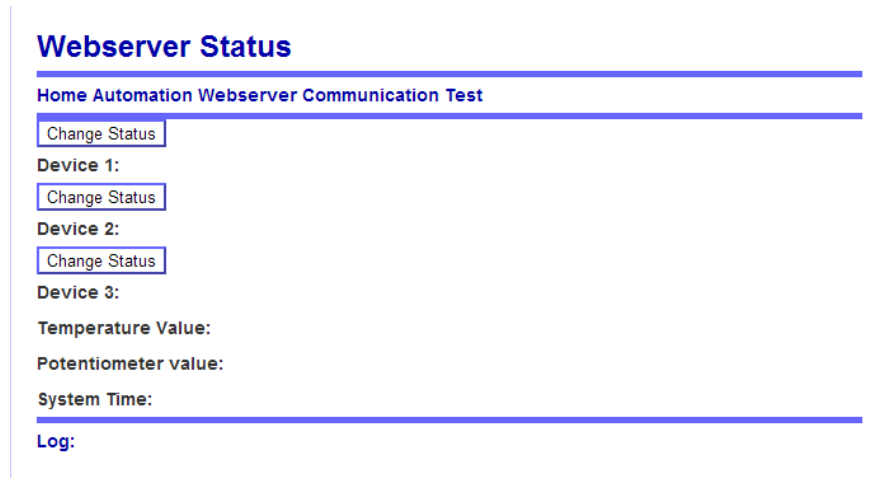


Figure 39 – Website template for home automation status

The “change status” push button has roles as inputs. Java code inside the web page allows these inputs to be sent and read by *cgi_sec_data ()*; in the MCU every time the button are pressed. These inputs are used to change the status of the LEDs in the home automation system. In the information display function, there are 6 data displays and data log functionalities which are used to display all devices status in home automation system. These functions display status of device 1, 2 and 3 (LEDs), status of temperature value, status of potentiometer value (ADC value), system time of the network / time since the system started, and the system activity log. The refresh rate for the website is set to 1 second. It means the website will read the CGI file (*secdata.cgi*) and update those values in every second.

4.3. LCD Touch Screen Program

The LCD module is programmed by using Workshop 4 IDE ViSi mode from 4D studio. This mode combines programming interface and visual design interface which provide flexibility and easiness to program. The first step is to design the visual looks for LCD module as a template display. The visual template itself is designed to provide 3 ON/OFF touch button, 1 normal touch button, 3 indicators represent the LED status, 1 system power indicator, and text box to provide text information. Figure 40 below shows the default display template for LCD home automation system module.

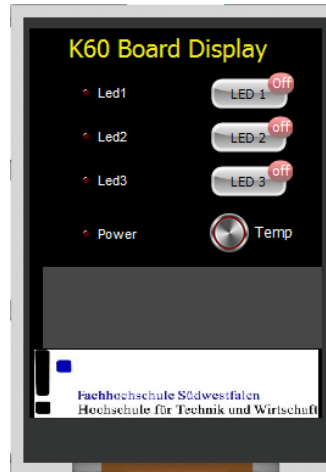


Figure 40 - Default display template for LCD module

In the programming part, LCD module is designed to have UART communication that will be used to communicate with MCU. Every time a touch button is pressed, a string command will be sent through this serial to MCU indicates that a touch button had been pressed. Next, LCD module will wait for the response from the MCU. This response then will be displayed as an LED indicator or display temperature value. The flowchart for the LCD module programming design is shown in Figure 41 below.

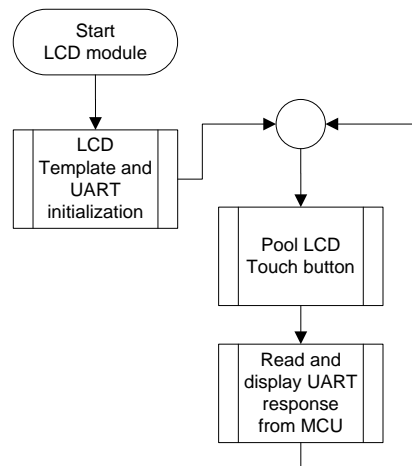


Figure 41 - LCD module main flowchart

LCD Template and UART Initialization

This sub-code is composed to initialize all of the functions that had been set in the default LCD display template such as, header text, touch buttons, LED and power indicators, text box, and so on.

Codes for initialization are generated automatically by the IDE follows the default template visual design itself. This sub code also includes initialization for UART communication that is used to communicate with MCU.

Pool LCD Touch Button

This sub-code is detecting whether there is a touch or non-touch condition in the touch buttons. When there is a touch, this code will check the touch status condition and send a string value to MCU through UART communication. The string value itself varied depends on which touch status condition occurs. Table 5 below shows the string value that LCD module sent to MCU depend on the touch status condition.

Table 5 - LCD module UART sends string value

Touch button	Status	String send
LED 1	OFF	“A0\r”
	ON	“A1\r”
LED 2	OFF	“B0\r”
	ON	“B1\r”
LED 3	OFF	“C0\r”
	ON	“C1\r”
Temp	Released	“RT\r”

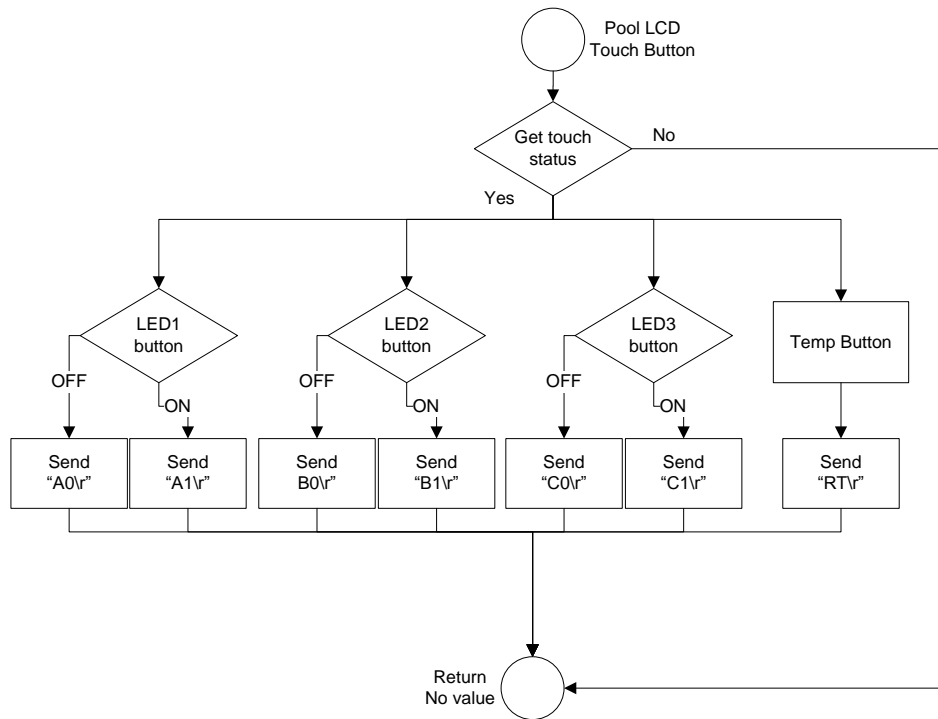


Figure 42 - Pool LCD touch button flowchart

Figure 42 shows how the Pool LCD Touch Button software flows. At first it checks if there is any touch condition in the LCD module. When there is a touch activity, it check on which button it pressed, and decide what kind of string data that need to be send to the MCU through UART communication.

Read and Display UART Response from MCU

Once LCD module sends the string value to the MCU, LCD module will check the status of UART receive data. This function will check is there any UART serial data comes from MCU as a response data. The response data itself are explained in table 3 at sub-chapter 4.1.3. When there is a response from MCU, for example “A1\r”, the system will check the first and second data string. Next, it will response based on this string data, on this case the system will set the indicator LED 1 in the display as ON. For temperature, when the LCD module sends “RT\r” to the MCU, MCU will response with string data with value “TXXXXXXXX\r” where T indicates type of data (temperature data) and XXXXXXXX is the temperature data itself. Temperature value then is displayed on the LCD. Flowchart for this sub-code can be seen in Figure 43.

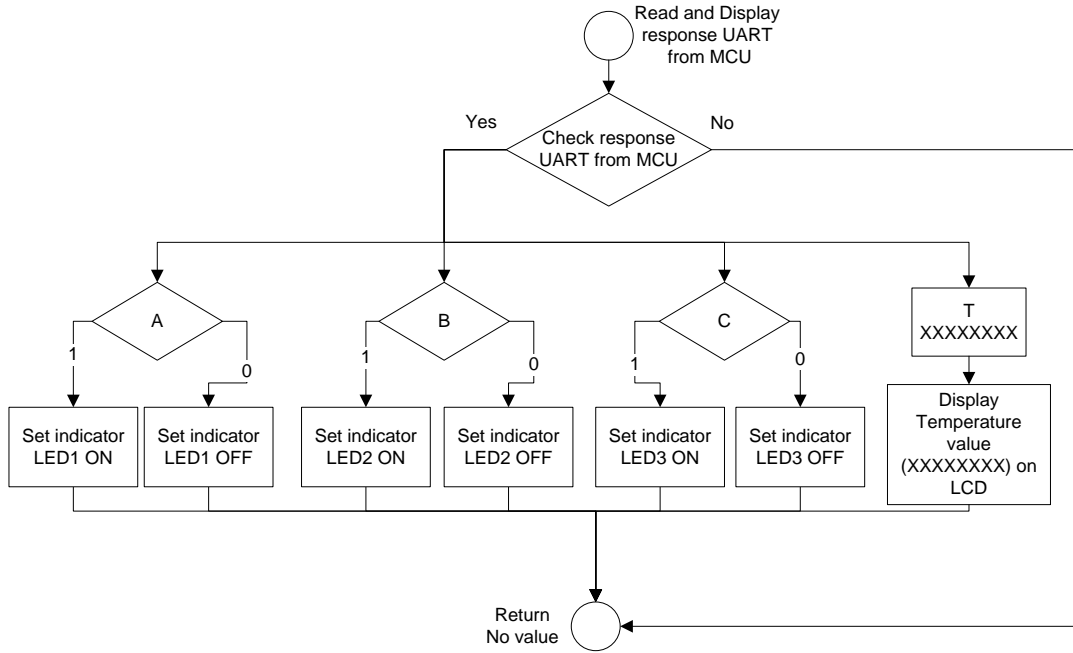


Figure 43 - Flowchart for read and display response UART from MCU

5. Assembly, Test, and Result

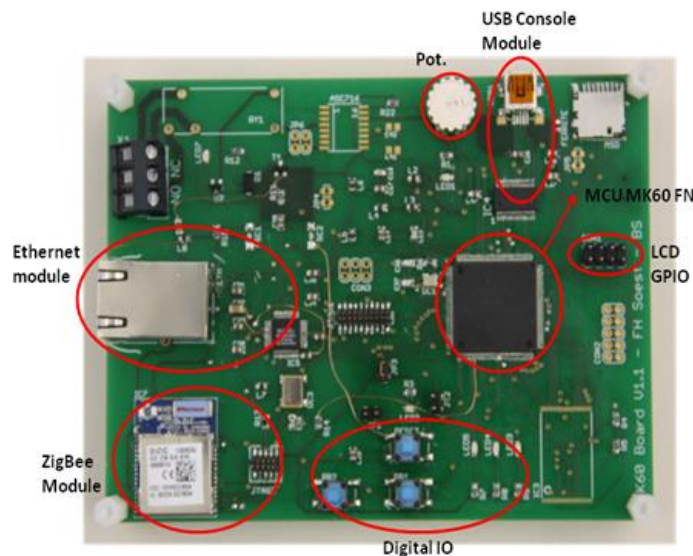
This chapter will discuss several processes related to hardware assembly process, functionality test including hardware and software, and the test result itself. Some obstacles which were encountered during development time are also explained here. Hopefully, with the lesson learned from this project, further development can be established with a better result.

5.1. Assembling process

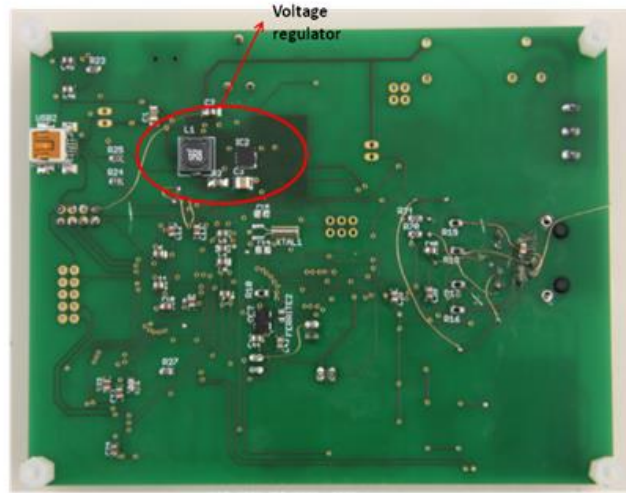
Due to the limitation of time, not all of the functions are applied and used in home automation custom board. From all of the functions, only main functionalities are applied on this custom board. There are 6 main functionalities in the home automation custom board which are:

1. Voltage regulator / power management circuit
2. Microcontroller circuit
3. Serial UART to USB IC Driver
4. Digital and Analog IO
5. ZigBee host circuit
6. GPIO UART 2 pin header connection for LCD module
7. Voltage regulator module

Figure 44 below shows all main components that already mounted to the home automation custom board.



(a)



(b)

Figure 44 - Board Assembly process top (a) and bottom (b)

Assemble process had been done simultaneously with the software test. However, on this sub-chapter, there will be more explanation about assembly process rather than software test. Software test will be explained in the next sub-chapter.

Assemble process start from the voltage regulator module. The voltage source itself comes from 5VDC USB1. As mentioned before, this module converts the 5VDC source to 3.3VDC by using step down TPS62111 IC. This assembly process goes first in order to make sure that there is no defect in voltage regulator output. Second assembly process is to mount the PK60FN MCU, digital IO module, MCU JTAG pin, and analog module (potentiometer). The idea for doing this process as a second step is to check the basic functionalities of MCU.

After MCU assemble and test process finish, the assemble process goes to Ethernet Module. This process is done in order to check whether the Ethernet module is working or defecting. Next assemble process is to mount the ZigBee module and LCD GPIO pin header. Afterwards, these modules are checked to test the UART communication functionalities between these devices and the MCU.

There are some modifications that need to be done for this custom board. These modifications are related to the mistake in board hardware design, mistakes in components order, and part of trial and error in test process.

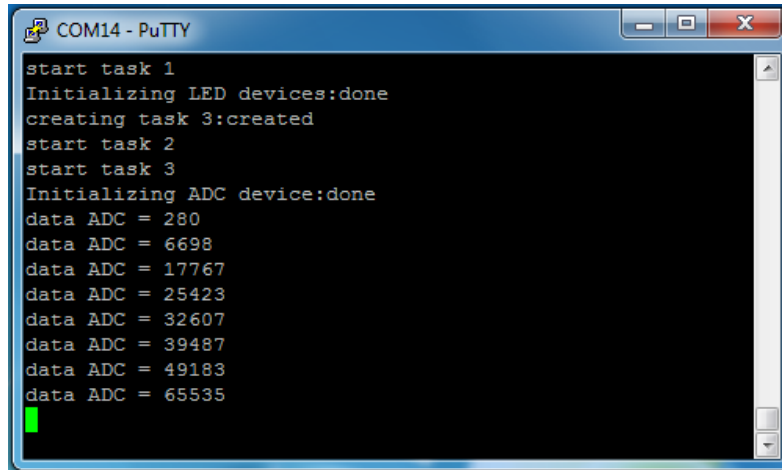
5.2. Functionality Test

There are several main functionality that is tested in this sub-chapter, these test are including MCU tasks, digital-analog IO, and Console test, UART communication test with LCD and ZigBee, Ethernet network test, and overall system test.

5.2.1. MCU tasks, digital-analog IO, and Console test

This test is conducted to check the basic functionality of the MCU by using push buttons in the board as digital inputs, LEDs as digital output, and potentiometer (internal ADC) as an analog input. This test is also conducted to check the stack and ProcessorExpert driver functionality in MQX RTOS. A test software scenario with 3 program tasks is made to do this functionality test. This First task of this test code is to create a system running indicator. This scenario use LED 3 in the control unit board to blink every 1 second to indicate that the system is running. Second task is used to read the digital inputs push button 1 and responds to these input conditions. Whenever PB1 is pressed, LED 4 will also turn to ON condition. The third task is coded to combine ADC function, push button, and console. In this task, push button 2 (PB2) is used to read the ADC value that connected to the potentiometer. Whether PB2 is pressed, ADC will measure potentiometer value then show it value to the console. Task 1 and task 2 are designed to start automatically after the MCU is booting up, meanwhile task 3 is generated by task 1 after the LED initialization. Task 1 has the highest priority, and then followed by task 2 and task 3.

Figure 45 below shows the live process of system execution on this test software. It shows step by step on which function is executed first. This console also shows the ADC measurement value from potentiometer whenever push button 2 is pressed. The data itself varied depend on the resistance given by the potentiometer.



```
COM14 - PuTTY
start task 1
Initializing LED devices:done
creating task 3:created
start task 2
start task 3
Initializing ADC device:done
data ADC = 280
data ADC = 6698
data ADC = 17767
data ADC = 25423
data ADC = 32607
data ADC = 39487
data ADC = 49183
data ADC = 65535
```

Figure 45 - Console response of the test software

From the response that appears in the UART 5 console and system break point in code warrior IDE, it shows that task 1 starts in the first place. Afterwards, it is followed by the starts of task 2, then return to task 1 for LED initialization and task 3 generation. After task 3 is created, system jump to execute task 3 main code, and then return back to task 1 to toggling the LED3. The process continues to check the status of push button 1 in the task 2 and response for it conditions. After the checking process, the execution goes to initialize ADC then followed by checking the status of push button 2 inside the task 3. This execution processes running continuously and alternately from task 1, task 2, and task 3. The execution length of each tasks is depend on the stack initialization for each tasks. Figure 46 below shows the illustration of these task processes.

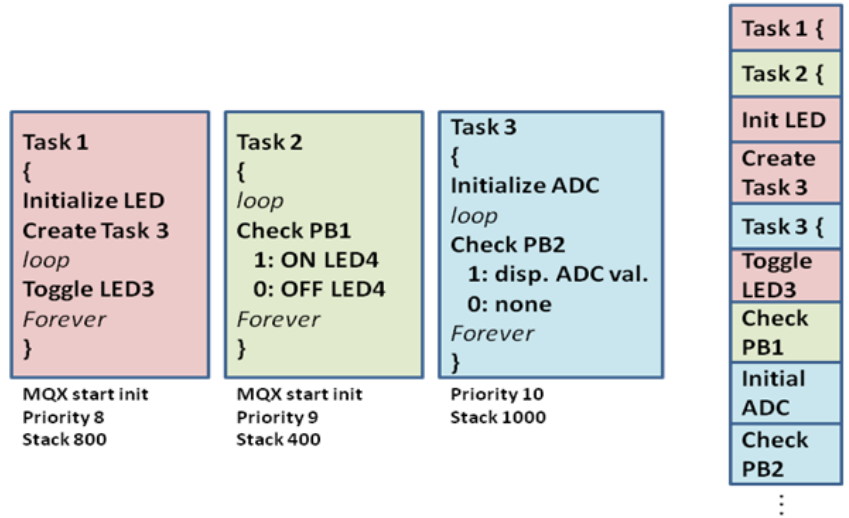
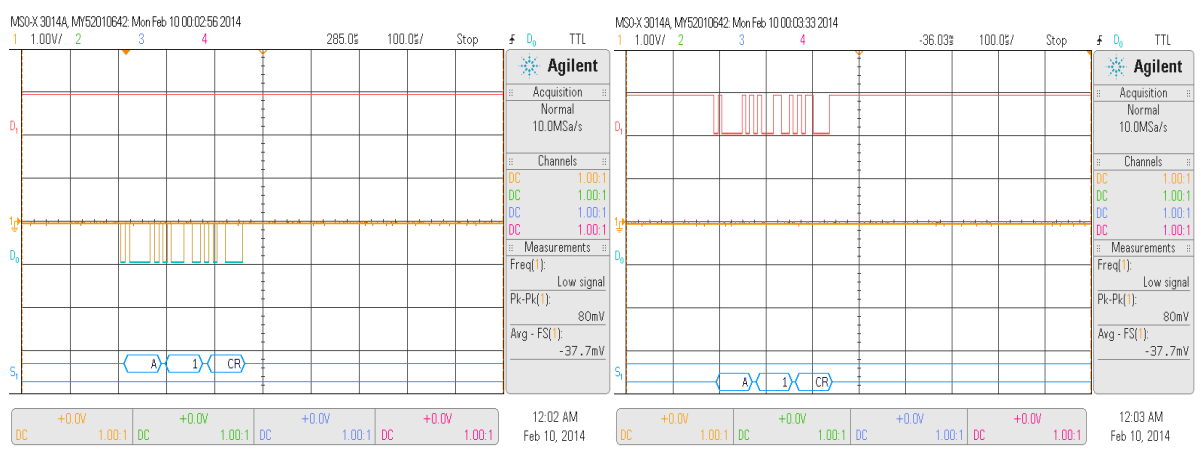


Figure 46 - Illustration of stacks process in the test software

5.2.2. LCD UART communication test

This sub-chapter will discuss about UART communication test between the LCD module and MCU PK60FN. LCD module will act as digital inputs with its touch buttons then send an indicator data whenever a touch button is pressed. MCU will read this UART data and response regarding to this data (LED ON/OFF). MCU will also send back this data as a feedback to the LCD module. This test is done by using digital oscilloscope to sniff the transmitted UART data.

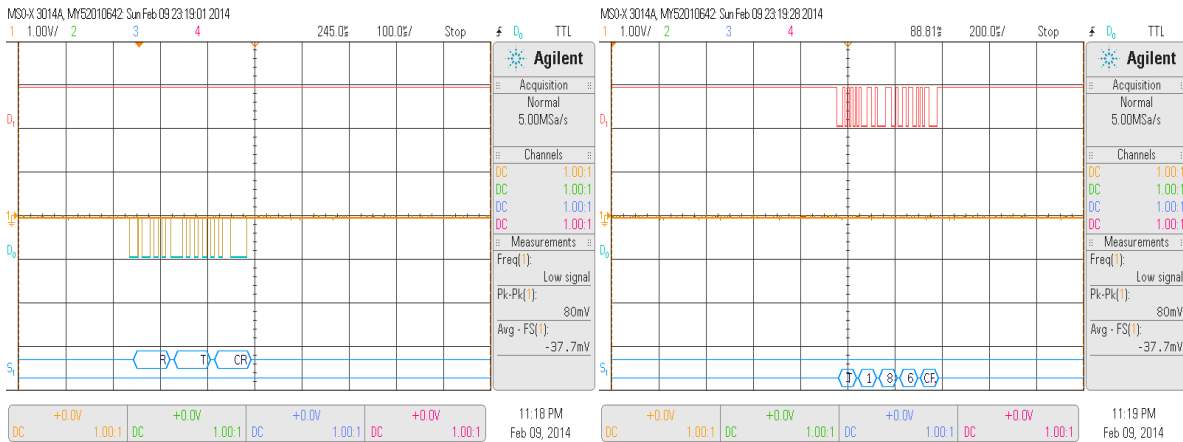


(a) (b)
Figure 47 - UART 2 data transmit (a) and receive (b) for touch button LED 1

Figure 47 above shows transmit (a) and receive (b) data when touch button LED 1 in LCD module is pressed from OFF to ON condition. LCD module will send string “A1\r” (figure 47(a)) to indicate

that this touch button had been pressed. MCU will response to turn ON the LED in the custom board. Subsequently, MCU will send the same string value “A1\r” (Figure 47(b)) as a feedback to LCD module. Feedback is used by LCD module to turn ON the LED indicator on the LCD module display. This condition and process are also applied for LED 2 and LED 3 touch button.

Different with the LED buttons, TEMP touch button will send a different string value which is “RT\r” whenever this touch button is pressed. This string indicates that LCD module is requesting the temperature value that measured from ZigBee node. As a reply, MCU will send “TXXX\r” to LCD module, where “XXX” is the temperature value. After LCD module receives this temperature data, this data will be displayed on the LCD. Figure 48 below shows the string values that transmitted and received when TEMP touch button is pressed.



(a)

(b)

Figure 48 - UART 2 data transmit (a) and receive (b) for touch button TEMP

Result for this simulation is shown in Figure 49 below. LED 1 touch button is in ON status, Led1 indicator is active, and text console shows the temperature value from the ZigBee sensor node.



Figure 49 - LCD module simulation for touch and data display

5.2.3. ZigBee UART communication test

There is some scenario conducted in ZigBee UART communication test. The tests itself are provided to check the communication between ZigBee host module with the MCU. For the first test, MCU will be replaced by CPU. CPU will pretend as MCU and communicate with the ZigBee host module through terminal console. This first test is conduct in order to check the data responses from this module. Several UART commands will be sent to the ZigBee host with one ZigBee node connected to it.

The first communication step is to check the communication line by sending “hi\n” command string. When ZigBee host receive this command, it replies with string “Hi\n”. Second command which is sent is “ls.nodes” string. This command is to check which and how many nodes are connected to the host. ZigBee node replies this command by sending a list of connected node. Maximum nodes connected are 15 nodes. On this test scenario, only one node is connected to the ZigBee host.

Next step is to request the temperature data to the ZigBee host module. This is done by sending “send 00000001 00000001 00000001\n” command string where the first 00000001 is the ZigBee node ID, followed by the second 00000001 which is temperature sensor ID, and the last 00000001 indicates command for request. Responds for this command is “DATA: 0x00000001 0x00000000 0x00000001 0x00000167\n” where first hex data indicates the ZigBee node ID, second hex indicates the host ID (always zero), third hex indicates the temperature sensor ID, and the last hex indicates

the raw temperature value. On this case, the raw temperature value is 167 in hexadecimal. Figure 50 below shows the UART data for transmit and receive in CPU terminal console.

```
hi
hi
Hi
lsnodes
ls .nodes
MODELIST: 0x0001 "SensNet-GPNode"
MODELIST: 0x0002 ""
MODELIST: 0x0003 ""
MODELIST: 0x0004 ""
MODELIST: 0x0005 ""
MODELIST: 0x0006 ""
MODELIST: 0x0007 ""
MODELIST: 0x0008 ""
MODELIST: 0x0009 ""
MODELIST: 0x000A ""
MODELIST: 0x000B ""
MODELIST: 0x000C ""
MODELIST: 0x000D ""
MODELIST: 0x000E ""
MODELIST: 0x000F ""
MODELIST: 0x0010 ""
send 00000001 00000001 00000001
DATA: 0x00000001 0x00000000 0x00000001 0x00000167
```

Figure 50 - First communication UART console data test

The second communication test is conducted by restarting the ZigBee node module. When this node restarts and boot, it will connect directly to the ZigBee host, initialized as the second ZigBee node. First ZigBee node status will remain as a dummy status. To check this status, another “ls.nodes” is send through UART communication. Next step is to check the temperature value from this second node. A string command is sent with value “send 00000002 00000001 00000001\n” to the ZigBee host module. As a response, ZigBee host send “DATA: 0x00000002 0x00000000 0x00000001 0x0000017E\n” where 17E hex is the temperature raw data.

There is another command to check all of the temperature data from the connected ZigBee nodes. This command is “send FFFFFFFF 00000001 00000001\n” where FFFFFFFF represents all the nodes, first 00000001 represents the temperature sensor ID, and second 00000001 represents request command. As a reply, ZigBee host only send one string data which is “DATA: 0x00000002 0x00000000 0x00000001 0x00000183\n”. This happens because only one node is connected which is ZigBee node 2. Figure 51 below shows this communication data from CPU terminal console.

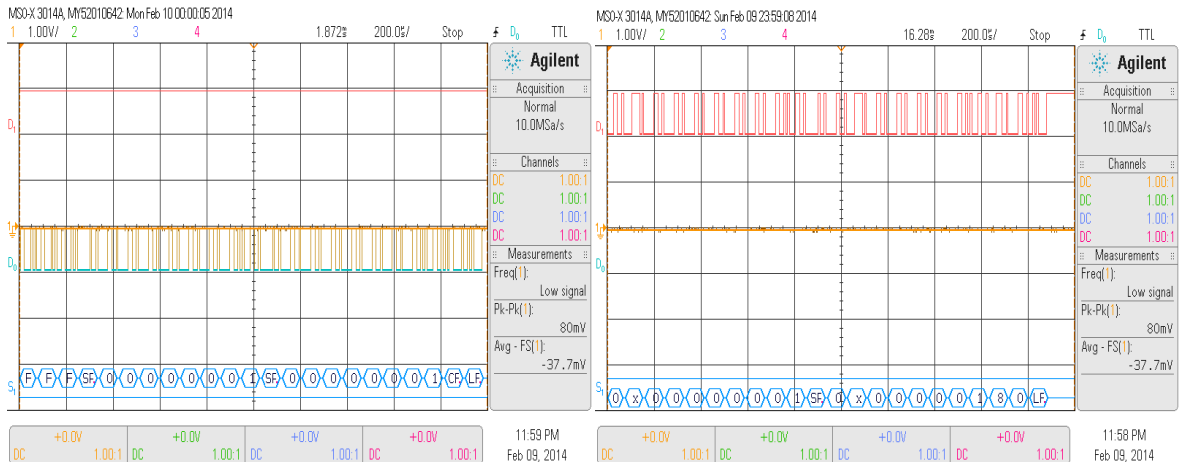
```

ls .nodes
MODELIST: 0x0001 "SensNet-GPNode"
MODELIST: 0x0002 "SensNet-GPNode"
MODELIST: 0x0003 ""
MODELIST: 0x0004 ""
MODELIST: 0x0005 ""
MODELIST: 0x0006 ""
MODELIST: 0x0007 ""
MODELIST: 0x0008 ""
MODELIST: 0x0009 ""
MODELIST: 0x000A ""
MODELIST: 0x000B ""
MODELIST: 0x000C ""
MODELIST: 0x000D ""
MODELIST: 0x000E ""
MODELIST: 0x000F ""
MODELIST: 0x0010 ""
send 00000002 00000001 00000001
DATA: 0x00000002 0x00000000 0x00000001 0x0000017E
send 00000001 00000001 00000001
send 0000FFFF 00000001 00000001
send FFFFFFFF 00000001 00000001
DATA: 0x00000002 0x00000000 0x00000001 0x00000183

```

Figure 51 - Second communication UART console data test

Third communication test is conducted to check the UART communication between the MCU and ZigBee node module. This is done by connecting the MCU UART 3 to ZigBee host module. An UART command is sent continuously every 4 second to request the temperature value. The command which is sent is “send FFFFFFFF 00000001 00000001\n”. As a response, ZigBee node sends the temperature value from the connected ZigBee node. A digital oscillator is used to sniff this UART transmit and receive data. The data is shown as follow in Figure 52.



(a)

(b)

Figure 52 - UART 3 data transmit (a) and receive (b) for touch ZigBee host module

On this master thesis project, the number of node that will be used in demonstration is only one node. Only this command that will be used for requesting temperature value due to the time limitation and simplify the software complexness.

5.2.4. Ethernet network test

On this sub-chapter, there will be a test scenario composed to test the functionalities of Ethernet module in the control unit home automation custom board. The software that will be used in this scenario is the example code provided by Freescale MQX [34]. The scenarios for this test are to plug the Ethernet cable with LAN access to the custom board, run the board, monitor the process through UART console, and check the website functionalities through a browser by accessing the IP address.

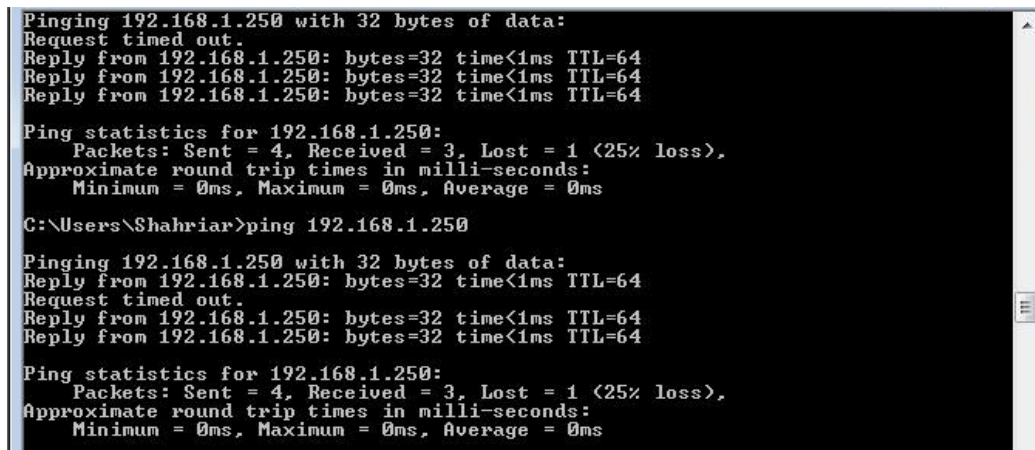
From the response that monitored from the console and windows command function. It seems like that the Ethernet module in this control unit board is not working properly. From several test, rarely the system can bind to the network and settle the IP address. When it connects, sometimes the connection is lost. When the system ping the IP address from windows command console, it shows some connections are lost.

```
Waiting for ethernet cable plug in ... Cable connected
DHCP bind ... Successful!
IP Address      : 192.168.1.250
Subnet Address  : 255.255.255.0
Gateway Address : 192.168.1.1
DNS Address     : 192.168.1.1
Waiting for ethernet cable plug in ... Cable connected
DHCP bind ... Successful!
IP Address      : 192.168.1.250
Subnet Address  : 255.255.255.0
Gateway Address : 192.168.1.1
DNS Address     : 192.168.1.1
Waiting for ethernet cable plug in ... Cable connected
DHCP bind ...
Waiting for ethernet cable plug in ... Cable connected
DHCP bind ...
Waiting for ethernet cable plug in ... Cable connected
DHCP bind ...
Waiting for ethernet cable plug in ... Cable connected
DHCP bind ...
```

Figure 53 - Ethernet initialization monitored from UART console

From figure 53, we can see that after trying the system for the second time, the system failed to bind the DHCP and get the IP address from the network. Sometimes it also loses its connection when the

system successfully connected and gets the IP address from the network. Mostly, from 4 ping process, only 3 replies come from the server. Figure 54 below shows the connection lost by ping the IP address from windows command console.



```
Pinging 192.168.1.250 with 32 bytes of data:
Request timed out.
Reply from 192.168.1.250: bytes=32 time<1ms TTL=64
Reply from 192.168.1.250: bytes=32 time<1ms TTL=64
Reply from 192.168.1.250: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.1.250:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\Users\Shahriar>ping 192.168.1.250

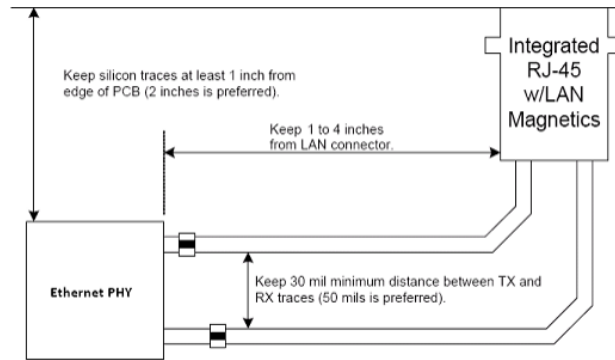
Pinging 192.168.1.250 with 32 bytes of data:
Reply from 192.168.1.250: bytes=32 time<1ms TTL=64
Request timed out.
Reply from 192.168.1.250: bytes=32 time<1ms TTL=64
Reply from 192.168.1.250: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.1.250:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

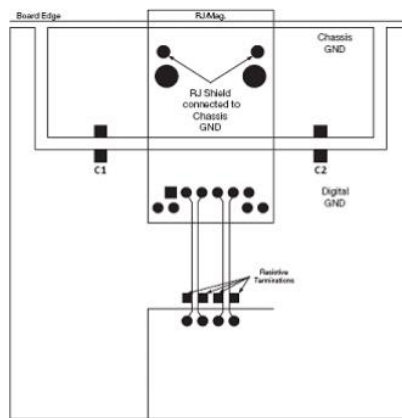
Figure 54 - Server ping test data from windows command console

Initially, in software and pin data connections point of view, Ethernet network module seems to do not have any crucial problem. In some test case, this module able to bind with the network and communicates. From this test process and the standard manual document, it is concluded that the faulty itself is more related to the hardware and external noise interferences. Several reasons can be concluded for this defective Ethernet module, these reasons are:

1. Ethernet driver is using MII communication mode with 100Base-TX for transmit and receive system [27]. It means that this system starts with a parallel to serial conversion, which converts data from the MAC into 125MHz serial bit stream. This high speed data transfer requires special handle on PCB design. Some of these requirements are [37, 38]: TX/RX pairs should be routed to the back of the integrated jack and away from the board edge. 49.9 ohm resistors should be placed within 400 mil of Ethernet PHY and should be placed next to the RX/TX pairs due to minimize stubs. These TX/RX line need to be routed over a continuous solid ground plane to maintain the impedance for the entire trace route. The Ethernet chassis itself should be connected to the separate ground which placed under the RJ45 jack.



(a)



(b)

Figure 55 - Specification of standard Ethernet layout [37]

2. Mistakes in ordering the component cause crucial faulty on RJ45 pin assignment and require a lot of line reworks. This rework gives possibility to the system to get interfere from many noises. Figure 56 below shows the reworks that had been made during the master thesis project development and test.

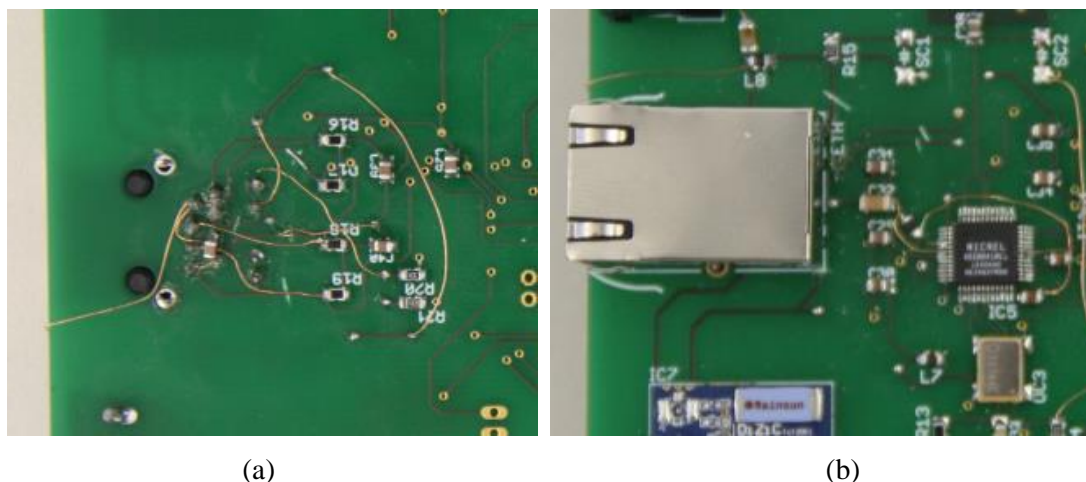


Figure 56 - Reworks made on the Ethernet module

A lot of work and minor modification had been done to try for resolve this problem. Some of these modifications are: add more passive capacitor in the signal TX/RX to the ground, this due to reduce the high frequency noises in the data. Change the RJ45 head connector to other type. Try to attach extra ferrite bead in series with the voltage source, this also to reduce the noises in the signal. Add extra 49.9 ohm resistances to handle with the impedances.

Some of these modifications give an extra performance to the Ethernet module, but still it is not perform well. Last option to handle this problem is by changing the hardware and PCB design following the standard requirements that are explained above.

In order to keep this master thesis on time track, Ethernet module is decided to be faulty and will be continued in the next development project. For whole system demonstration, TWRMK60FN development board will be used and attached with LCD and ZigBee functionalities. Some special setups are made to support this development board, such as BSP/PSP setup and UART 1 and UART 4 assignments.

5.2.5. Web server and overall system test

As mentioned in the previous chapter, web server and overall system test will be conducted by using TWRMK60FN development board. ZigBee host module and LCD module will be attached to this module through side connector in primary tower board. Figure 57 below shows how TWRMK60FN development module is attached other modules.

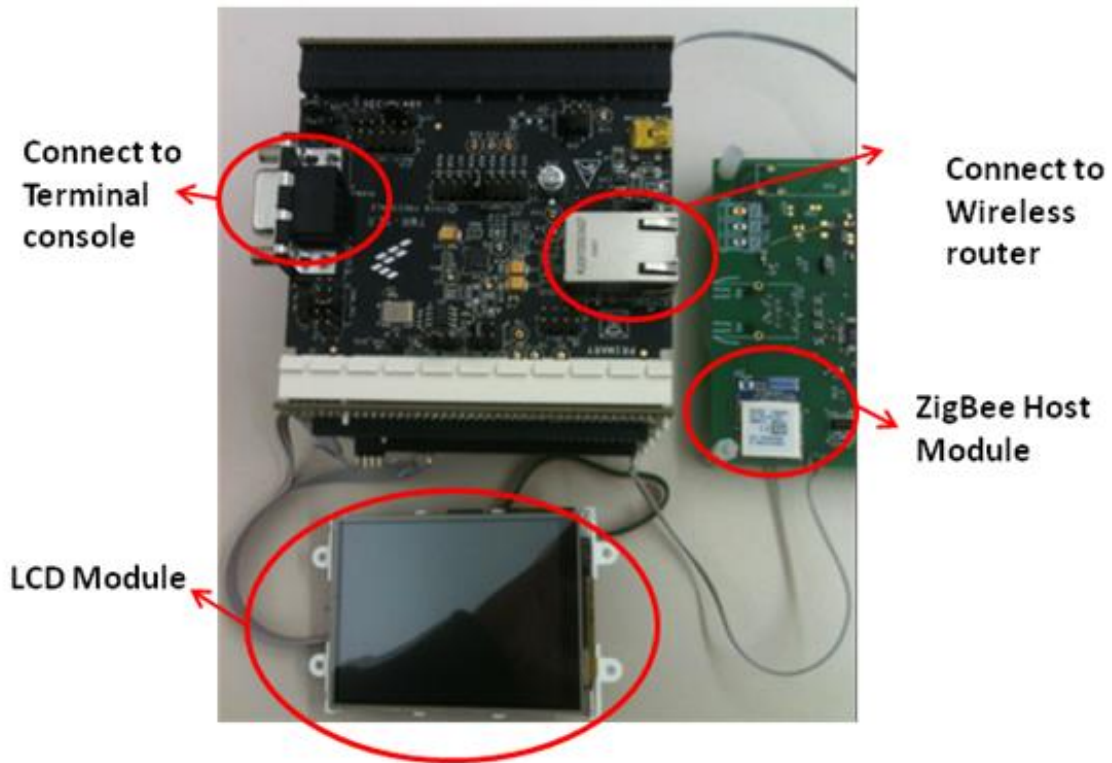


Figure 57 - TWRKM60FN with ZigBee node and LCD module

When the MCU system is started, system will automatically start the 1st task and initialize of all system including network initialization. This initialization includes GPIO initialization (LEDs), UART 1 and UART 4 initialization, ADC initialization, and Network initialization. Once the system successfully binds with the network and gets the IP address, the website now can be accessed through this IP address within the same network area. All the IO functionality can also be tested. First test is to check the functionalities of LED buttons from LCD module. MCU able to read the LED touch buttons through UART, update this status to web server and UART console, send echo to LCD module, and drive the LEDs in the board. Figure 58 below shows the test system scenario includes initialization and status data in UART console.


```
COM15 - PuTTY
init GPIO
init GPIO OK
init UART 2 and UART 3:
UART 2 open (LCD communication)
UART 3 open (ZigBee communication)
init ADC
ADC device open

Waiting for ethernet cable plug in ...
Cable connected

DHCP bind ... Successful!

IP Address      : 172.19.65.39
Subnet Address  : 255.255.255.0
Gateway Address : 172.19.65.2

DNS Address     : 195.37.252.20
DEVICE 1 = ON
DEVICE 1 = OFF
DEVICE 2 = ON
DEVICE 2 = OFF
DEVICE 3 = ON
DEVICE 3 = OFF
DEVICE 1 = ON
DEVICE 1 = OFF
DEVICE 2 = ON
DEVICE 3 = ON
DEVICE 3 = ON
DEVICE 1 = OFF
DEVICE 2 = ON
DEVICE 3 = ON
requesting Temp Value to ZigBee
temperature value = 180
requesting Temp Value to ZigBee
temperature value = 180
requesting Temp Value to ZigBee
temperature value = 180
```

Figure 58 - System test scenario UART console

Second test is to check the press button from the website. To check this website, a browser is used to open the IP address of this webpage which is “172.19.65.39”. There is some lagging phenomena happen when website is accessed for the first time. Normally it does not shows the home automation system status data from CGI file until buttons in the website are pressed for three times in random. Once the website works, now it able to send and update the push button information to the system status and change the LEDs status in the board. It also sends this status to console as monitoring purpose. This website also able to read home automation status data from CGI file and display this data to the website. This data includes LEDs status data, potentiometer data, and temperature data.

Webserver Status

Home Automation Webserver Communication Test

Device 1: Off

Device 2: On

Device 3: On

Temp Detection: 385

Garage Door: 43

System Time: 00:36:17

Log:

```
00:31:36 Device3 ON
00:31:35 Device2 ON
00:31:33 Device1 OFF
00:31:28 Device3 ON
00:31:27 Device2 ON
00:31:26 Device1 OFF
00:31:10 Device1 ON
00:31:07 Device3 OFF
00:31:06 Device3 ON
00:31:05 Device2 OFF
```

(a)



The screenshot shows a web browser window with the address bar containing "172.19.65.39/secdata.cgi". Below the address bar, there is a language selection dropdown set to "Englisch" and a button labeled "Soll sie übersetzt werden". The main content area displays the following text:

```
Off
On
On
00:33:54
00:31:36 Device3 ON
00:31:35 Device2 ON
00:31:33 Device1 OFF
00:31:28 Device3 ON
00:31:27 Device2 ON
00:31:26 Device1 OFF
00:31:10 Device1 ON
00:31:07 Device3 OFF
00:31:06 Device3 ON
00:31:05 Device2 OFF
46
384
```

(b)

Figure 59 - Webpage (a) and CGI data (b) for home automation system test

Last test is to request the temperature value from LCD module and display it back to the LCD. When the TEMP touch button in LCD module is pressed, the MCU send the temperature value data to the LCD and shows this activity to the UART console. LCD then displays this temperature value. Home automation system in the MCU is set to request the temperature value to ZigBee host every 4 seconds. Whenever the temperature data is not available, MCU will read an error value and shows this error as “536805908” temperature value.

5.3. Problem Faced and Lesson Learned

There are many obstacles and lessons that learned from this home automation thesis project. These obstacles mostly came from Ethernet network hardware, CodeWarrior 10.3 beta version IDE, and learning sources. These obstacles and lessons learned are explained more below.

- Project complexity with time limitation

At the beginning of this project development, this home automation project seems not to be really complex. Some features had already done in the previous study such as ZigBee host and node network, current sensors application, and web server example. By the time project run, some additional functionalities are requested to add to the control unit home automation board such as 2nd USB module, uSD module, and so on. This project grows its complexity in terms of hardware and software development part.

The first month was finished to study all the literature data, try to familiarize with the home automation system, learned the IDE, and custom board components selection. Second month was finished to learn the system module, designing the schematic, designing the PCB layout, and software development for LCD module and web server. By the end of the second month, which still remain with many high load, this project was reduced to only have the main home automation functionalities part.

The third month is dedicated to start the board assembly and test process. The assembly itself took 2 weeks due to some crucial problems that appears in the middle of this process. This problem was made the project delayed for 1 week. Some problem are not solved, pushed the project to find another alternative and solutions. For the software development and test itself, it took more than 1 week of work includes the weekend. Some obstacles were also found here regarding to the CodeWarrior IDE and the MQX RTOS development software.

The fourth month is dedicated for writing report, documentation, and presentation. Once the main system worked, report and documentation is started. The report and documentation itself planned to be done within 3 weeks and the last week will be dedicated for presentation preparation.

From all of this process and project experience, 4 months for doing this entire home automation project is almost impossible, except the person within the project has plenty experiences in this field and has a lot of technical support.

- Research community

Freescale MCU is not really commonly used in the embedded system developer community compared to Texas instrument, Atmel, or ST semiconductor. There are not so many information can be found about problem and solving on the Freescale community webpage. Freescale forum members are not really active even there are Freescale engineer inside this forum. Questions regarding to some problems can take weeks even month to answer. Personal message to the Freescale engineer is better on this case. Big self curiosity, motivation, and efforts are needed to dig into many sources and try to find the problem solutions.

- Hardware design

In this hardware design part, Ethernet module is failed to be operate perfectly. This is due to lack experiences in designing and developing the PCB, especially when it is related to high speed data transmission. Many crucial and important parts ignored when designing this PCB. For example, the importance of ground field, data impedance measurement, components selections and so on. Some hardware modules are also failed to be implement in this home automation custom board. This is more to the lack of project time which only reserved for 1 month to design the complete schematic and PCB for home automation system.

- Software design

Problem faced in software design is more to unreliable CodeWarrior 10.3 beta version. The project faced a lot of bugs when designing the home automation software. Some of these errors are: unable to create MQX project from scratch, some ProcessorExpert drivers are failed to build with this IDE version such as BIT IO driver and Term driver, and many small cases. Another case is the lack of informations that can be gathered regarding to the basic usage and function of CodeWarrior IDE, MQX libraries, and ProcessorExpert drivers. Some examples are given by the library, but still some modification and changes are needed to adjust this library to be suit with home automation software system. These learning processes easily take time up to one week just to find out and understand a system library or setup.

The unreliable CodeWarrior 10.3 beta version problems are mostly solved in the newest version of CodeWarrior 10.5. The problem is the free version of this IDE is limited to 64Kb code size meanwhile home automation system software requires RTCS real time library for running the web server and consumes code size more than this limit.

CodeWarrior is a very powerful IDE which provide a high level of programming in the top of MQX operating system. Many drivers are prepared by this IDE and MQX RTOS library. These provide a very flexible and easy programming method for the programmer. Programmers only need to set up the driver once based to the functionalities and processor that he want to use, then use this drivers to drive all the processor functionalities. With this CodeWarrior 10.5, developing home automation project software can be much easier because it does not require the programmer to dive inside and develop their own functions initialization and driver.

In the time this paper is made, author get a CodeWarrior 10.5 IDE evaluation version which provide full functionality without limitation of code size. The problem is only that this IDE works only for 30 days. Furthermore, the project itself already in the ending phase which only provide 2 weeks of time to complete the documentation and presentation project.

6. Conclusion and Future Works

This thesis has presented a basic functionality of central control unit for home automation system supporting ZigBee and Wi-Fi network. The system is able to read information (temperature) from ZigBee node through ZigBee host and display this information to the website. This system is also able to drive LEDs (on and off) as media to simulate home devices. With these basic functionalities, a simple home automation system can be applied to support elderly and impaired people to control their home appliances.

Although the basic home automation system functionality is completed, there are still a lot of work should be done on this project. One of them is regarding to the data communication with ZigBee host module. Future work, MCU should be able to read not only the temperature data, but also other functionalities in the node sensor (humidity, actuator, and relay). Furthermore, MCU also should be able to access more than one ZigBee node. Other improvement that needs to be mention in the software part is the status display system, both in the website and LCD module. This system should be able to shows all the sensor data that gathered from several sensor nodes to the website and LCD module. Some security system also needs to be applied to the network in order to protect this system from hijack.

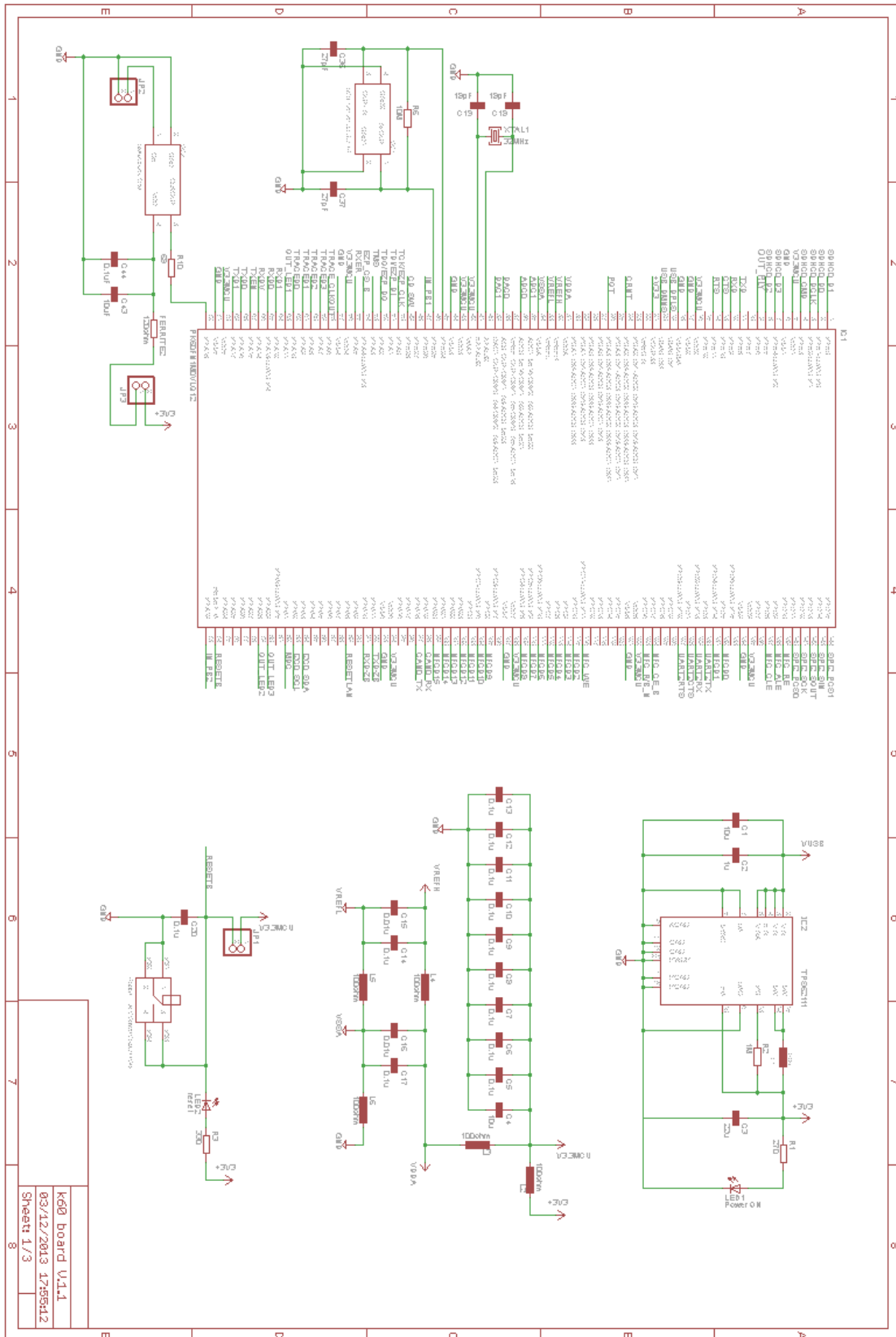
On the hardware side, some failures are discovered in this project during the test period. Major failure is happened in the Ethernet module part. Lack of knowledge in PCB design for high speed data transmission and limited project time had become the main reasons for this failure. Author suggest for future work to change this Ethernet module into direct Wi-Fi module in order to simplify the system itself. Other future work suggestion in hardware parts is to remove unnecessary components and modules in the final hardware design. This suggestion is given to minimize the PCB size, design, and costs. Four layers PCB with independent ground and VCC layers might be needed in order to handle the noise and signal disturbances.

Appendices

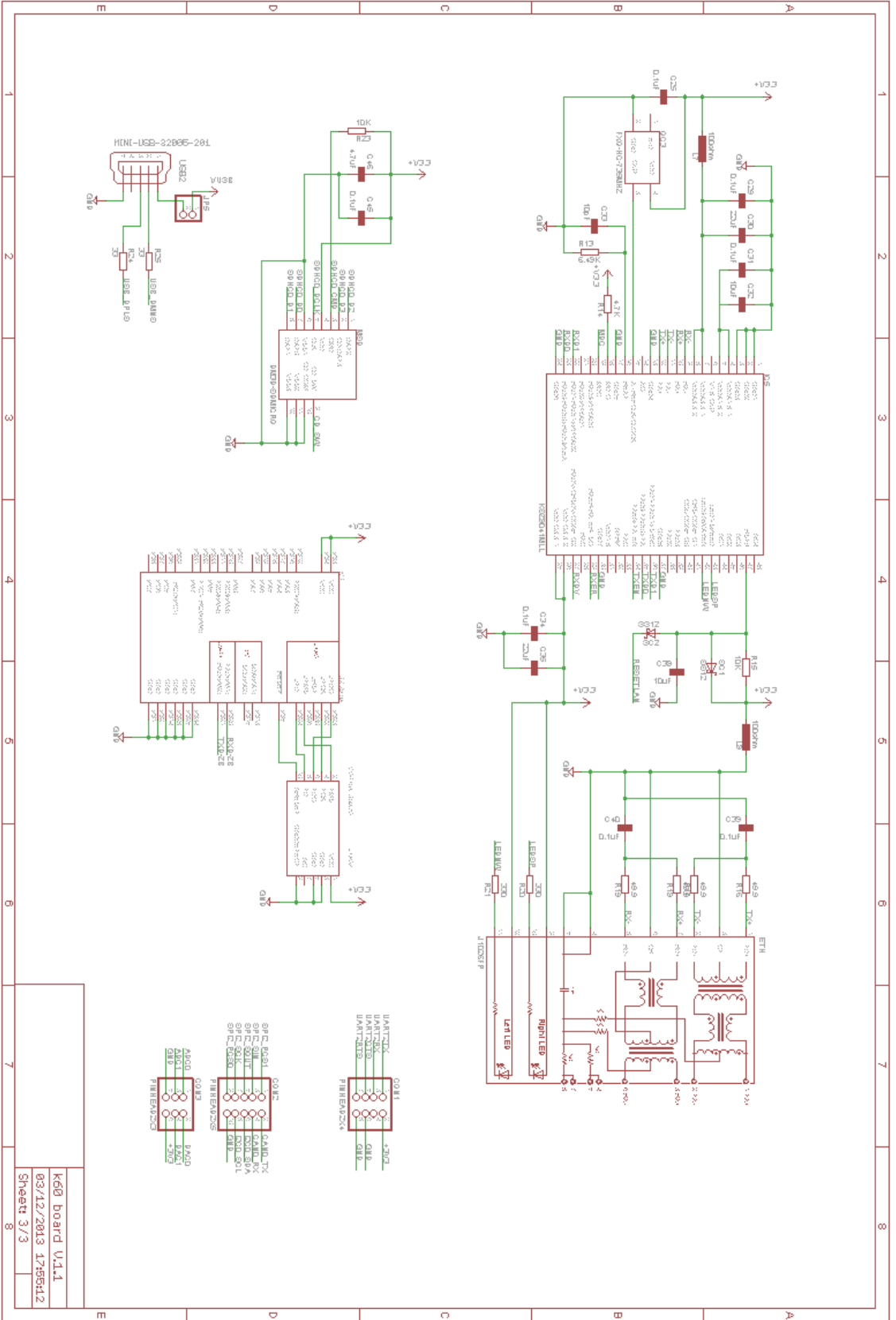
All the materials relevant to this thesis are provided as softcopy in DVD-ROM. The contents of the DVD-ROM are as follows:

- Software Code for Home Automation System
- Software Code for LCD Module
- Control Unit Schematic and PCB designs
- Important documents include datasheets, Application notes, and so on.

Schematic of Home Automation Control Unit



K80 board V1.1.1
03/12/2013 17:55:12
Sheet 1/3



K660 board V1.1
 03/12/2013 17:05:12
 Sheet 3/3

List of Figures

Figure 1 - Home automation system in general [2].....	3
Figure 2 - Master thesis project Gantt-chart	5
Figure 3 - ZigBee mesh and bridging network [10].....	9
Figure 4 - Designed home automation system.....	10
Figure 5 - Front side (a) and back side (b) of TWR-K60F120M development board [11].....	11
Figure 6 - Four stack logic status (a) and usage illustration of tasks (b) [17]	13
Figure 7 - Freescale software development level [19] (a) and Illustration of BSP / driver configuration (b)	15
Figure 8 - Simplified ZigBee module block diagram [21].....	16
Figure 9 - uLCD-32PTU LCD and touch module [22].....	17
Figure 10 - schematic and PCB design with Eagle from CadSoft	21
Figure 11 - Power supply management circuit schematic	22
Figure 12 - Functional assignment of MCU PK60FN	23
Figure 13 - Ethernet IC driver schematic design	25
Figure 14 - ZigBee module schematic diagram	26
Figure 15 - Serial UART 232 to USB converter circuit diagram.....	26
Figure 16 - Relay and current sensor circuit diagram.....	27
Figure 17 - Example of 45 degrees turned signal lines.....	28
Figure 18 - Top layer modules placement	29
Figure 19 - bottom layer module placement	30
Figure 20 – Home automation system software tasks diagram.....	31
Figure 21 - BSP setup files	32
Figure 22 - <i>user_config.h</i> : UART 2 and UART 3 configuration.....	33
Figure 23 - <i>init_gpio.c</i> : UART3 pin assignments	34
Figure 24 - <i>twrk60f120m.h</i> : UART mode initialization.....	35
Figure 25 - <i>serial.h</i> : options for UART mode initialization.....	35
Figure 26 - Web server/main task flowchart.....	36
Figure 27 - GPIO initialization flowchart	37
Figure 28 - <i>_task_create</i> function	38
Figure 29 - RTCS setup processes [35]	38
Figure 30 - Web server initialization flowchart	39
Figure 31 - RTCS server data flow	40
Figure 32 - Flowchart for <i>cgi_sec_data()</i> ;	41
Figure 33 - IO task data flow	42
Figure 34 - IO task flowchart.....	43
Figure 35 - Pool UART 2 Input flowchart.....	44
Figure 36 - Flowchart of <i>SEC_SetDevice1_ON()</i> ;	45
Figure 37 - Flow chart of read ADC function.....	46
Figure 38 - Flowchart of read ZigBee temperature value	48
Figure 39 – Website template for home automation status.....	49

Figure 40 - Default display template for LCD module	50
Figure 41 - LCD module main flowchart.....	50
Figure 42 - Pool LCD touch button flowchart	52
Figure 43 - Flowchart for read and display response UART from MCU	53
Figure 44 - Board Assembly process top (a) and bottom (b)	55
Figure 45 - Console response of the test software	57
Figure 46 - Illustration of stacks process in the test software	58
Figure 47 - UART 2 data transmit (a) and receive (b) for touch button LED 1	58
Figure 48 - UART 2 data transmit (a) and receive (b) for touch button TEMP.....	59
Figure 49 - LCD module simulation for touch and data display.....	60
Figure 50 - First communication UART console data test	61
Figure 51 - Second communication UART console data test.....	62
Figure 52 - UART 3 data transmit (a) and receive (b) for touch ZigBee host module	62
Figure 53 - Ethernet initialization monitored from UART console	63
Figure 54 - Server ping test data from windows command console	64
Figure 55 - Specification of standard Ethernet layout [37].....	65
Figure 56 - Reworks made on the Ethernet module.....	66
Figure 57 - TWRKMK60FN with ZigBee node and LCD module	67
Figure 58 - System test scenario UART console	68
Figure 59 - Webpage (a) and CGI data (b) for home automation system test	69

List of Tables

Table 1 - KSZ8041MLL MII signal definition [26]	19
Table 2 - Functionality pin assignment for MCU PK60FN1M0VLQ12	23
Table 3 - Received string values from LCD module	43
Table 4 - ZigBee command and the data response	46
Table 5 - LCD module UART sends string value.....	51

Acronym and Abbreviation

LED	Light Emitting Diode
LCD	Liquid Crystal Display
GSM	Global System for Mobile communication
NWK	Network Layer
PCB	Printed Circuit Board
USB	Universal Serial Bus
AC Network	Alternating Current network
PDA	Personal Digital Assistance
IP	Internet Protocol
LON-Bus	Local Operating Network-Bus
BAC net	Building Automation and Control network
EIB - KNX	European Installation Bus –KNX (standard)
MCU	Microcontroller
TWR-SER	Tower-Serial module
SDHC	High Capacity SD memory card
LQFP	Large Quad Flat Package
SPI	Serial Peripheral Interface
ADC	Analog to Digital Converter
DHCP	Dynamic Host Configuration Protocol
UART	Universal Asynchronous Receive Transmit
CAN	Controller Area Network
I2C	Inter Integrated Circuit
I2S	Inter IC Sound
USB OTG	USB On The Go
JTAG	Joint Test Action Group
IDE	Integrated Development Environment
RTOS	Real Time Operating System
BSP	Board Support Package
PSP	Project Support Package
GPIO	General Purpose Input Output
RTCS	Real Time TCP/IP Communication Suit
MFS	MS-Dos File System
MS Dos	Microsoft Disk Operating System
TCP/IP	Transmission Control Protocol/Internet Protocol
RF	Radio Frequency
IEEE	Institute of Electrical and Electronics Engineering
PPP	Point to Point Protocol
CSS	Channel Stacking Switches
AES	Advanced Encryption Standard
IC	Integrated Circuit
MII	Media Independent Interface

PHY	Physical layer Device
MDIO	Management Data Input Output
uSD	Micro SD memory card
NC	Not Connect
TXD	Transmit Data
RXD	Receive Data
RTS	Request To Send
CTS	Clear To Send
VDC	Volt of Direct Current
VCC	Voltage Source
CGI	Common Gateway Interface

References

- [1] A. Venkatesh, CRITO, “Smart Home Concepts: Current Trends”, California, USA: University of California. Retrieved from <http://www.crito.uci.edu/noah/paper/smarthome.pdf>
- [2] LMA, “Software Personalizzati”, 2014. Retrieved from <http://www.lma-to.it/pagina.php?pag=software>
- [3] Renato J. C. Nunes and Jose C. M. Delgado, “An Internet Application for Home Automation”, Lisbon, Portugal: Technical University of Lisbon, 10th Mediterranean Electrotechnical Conference IEEE, 2000.
- [4] A. Z. Alkar, et al, “IP Based Home Automation System”, Ankara, Turkey: Hacettepe University, IEEE Transaction on Consumer Electronic, Vol. 56, No. 4, 2010.
- [5] N. Sriskanthan*, F. Tan, and A. Karande, “Bluetooth based home automation system”, Singapore: Nanyang Technological University, 2002.
- [6] A. Alheraish, “Design and Implementation of Home Automation System”, IEEE Transactions on Consumer Electronics, vol. 50, no. 4, pp. 1087-1092, Nov. 2004.
- [7] ZigBee Standards Organizations, “ZigBee Specification: ZigBee Document 053474r17”, 2007. Retrieved from http://people.ece.cornell.edu/land/courses/ece4760/FinalProjects/s2011/kjb79_ajm232/pmeter/ZigBee%20Specification.pdf
- [8] J. Graafmans, V. Taipale, and N. Charness, “Gerontechnology: A Sustainable Investment of the Future”, Amsterdam, The Netherlands: IOS, 2007.
- [9] A. Z. Alkar, et al, “Web Based ZigBee Enabled Home Automation System”, Ankara, Turkey: Hacettepe University, 13th international conference on Network-based Information System, IEEE, 2010.
- [10] Aware Point Blog, “ZigBee wireless technology for hospital”, Retrieved from <http://www.awarepointblog.com/2010/06/zigbee-wireless-technology-for-hospital.html>
- [11] Freescale Semiconductor, “Quick Start Guide: TWR-K60F120M”, Retrieved from http://cache.freescale.com/files/32bit/doc/quick_ref_guide/TWRK60F120MQSG.pdf
- [12] Freescale Semiconductor, “Data Sheet: K60 sub-family, K60P144M120SF3”, Retrieved from http://cache.freescale.com/files/microcontrollers/doc/data_sheet/K60P144M120SF3.pdf
- [13] Freescale Semiconductor, “CodeWarrior Development Studio for Microcontroller V10.5”, Retrieved from http://cache.freescale.com/files/soft_dev_tools/doc/support_info/CW_MCU_10-5_RN.pdf?fp=1
- [14] Eclipse, Retrieved from www.eclipse.org
- [15] Freescale Semiconductor, “Freescale MQX™ RTOS – User Guide”, Retrieved from http://cache.freescale.com/files/32bit/doc/user_guide/MQXUG.pdf
- [16] Freescale Semiconductor, “Using MQX Libraries”, Retrieved from http://cache.freescale.com/files/32bit/doc/app_note/AN3907.pdf

- [17] Chung-Ta King, “task and scheduling”, Taiwan: Department of Computer Science National Tsing Hua University.
- [18] Freescale Semiconductor, “Processor Expert Software and Embedded Components”, Retrieved from http://www.freescale.com/webapp/sps/site/homepage.jsp?code=BEAN_STORE_MAIN
- [19] Freescale Semiconductor, “MQX: Freescale MQX Software Solutions”, Retrieved from http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=MQX
- [20] DiZiC, “DiZiC DZ-ZB-Sx RF Module”, Retrieved from http://www.signal.com.tr/pdf/cat/Datasheet_DZ-ZB-SA.pdf
- [21] TEM catalogue, RF Communication Module, Retrieved from http://www.tme.eu/en/katalog/rf-communication-modules_113194/
- [22] 4D System, “3.2” microLCD PICASO Display μ LCD-32PTU”, Retrieved from <http://www.4dsystems.com.au/downloads/microLCD/uLCD-32PTU/Docs/uLCD-32PTU-Datasheet-REV1.8.pdf>
- [23] 4D system, “Workshop 4 User Guide”, Retrieved from <http://www.4dsystems.com.au/downloads/Software/4D-Workshop4-IDE/Docs/Workshop-4-User-Guide-REV1.3.pdf>
- [24] D. Bhadra, Vikas S. Vij, Kenneth S. Stevens, “A Low Power UART Design Based on Asynchronous Techniques”. Utah, USA: University of Utah.
- [25] FTDI Technology, “FT232R USB UART IC”, Retrieved from http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf
- [26] Micrel, “KSZ8041TL/FTL/MLL Data Sheet”, Retrieved from http://www.micrel.com/_PDF/Ethernet/datasheets/ksz8041tl-ftl-ml.pdf
- [27] Allegro, “ACS716 Data Sheet”, Retrieved from <https://www.google.de/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CDEQFjAA&url=http%3A%2F%2Fwww.allegromicro.com%2F~%2Fmedia%2FFiles%2FDatasheets%2FACS716-Datasheet.ashx&ei=PQv9UoDsJsiThgDuIEI&usg=AFQjCNEk-k-3qrWnp8pVR2gRTwn-4zKFtg&bvm=bv.61190604,d.ZG4&cad=rja>
- [28] CadSoft, “Eagle PCB Software”, Retrieved from <http://www.cadsoftusa.com/eagle-pcb-design-software/product-overview>
- [29] Texas Instrument, “TPS62110, TPS62111, TPS62112, TPS62113 Data Sheet”, Retrieved from <http://www.ti.com/lit/ds/symlink/tps62111.pdf>
- [30] Freescale Semiconductor, “TWR-K60F120M Tower Module User’s Manual”, Retrieved from http://cache.freescale.com/files/microcontrollers/doc/user_guide/TWRK60F120MUM.pdf
- [31] Fairchild, “BSS138 Data Sheet”, Retrieved from <http://www.adafruit.com/datasheets/BSS138.pdf>
- [32] Pololu Robotics and Electronics, “Pololu basic SPDR Relay Carrier 5VDC”, Retrieved from <http://www.pololu.com/product/2480>
- [33] Freescale Semiconductor, “Security Web Server demo code”, Retrieved from C:\Freescale\Freescale MQX 3.8\demo\security_webserver

- [34] Freescale Community, “K60 UART Problems”, Retrieved from <https://community.freescale.com/message/311704#311704>
- [35] Freescale Semiconductor, “AN3907: Using MQX Libraries”, Retrieved from http://cache.freescale.com/files/32bit/doc/app_note/AN3907.pdf
- [36] R. Stemmer, “General Purpose Wireless Sensor Node for Home Automation”, Soest, Germany: FH Suedwestfalen Soest, 2014.
- [37] Freescale Semiconductor, “AN4215: i.MX28 Layout and Design Guidelines”, Retrieved from http://cache.freescale.com/files/32bit/doc/app_note/AN4215.pdf
- [38] Rabbit, “TN266: PCB Layout for Ethernet PHY Interface”, Retrieved from http://ftp1.digi.com/support/documentation/022-0137_F.pdf