Monitoring and logging system for home use.

An 8-bit approach.

Dmitri Peredera
Supervisor: Mikael Ekström

5/10/2011
Abstract.

The thesis discusses the creation and implementation of a monitoring and notification system for home use. The topic may have interest during the warm Swedish summer and long winter nights, when the weather conditions put home flora to the test. The vivid imagination together with a life-long experience of the author predicts a total devastation of green plants inside the apartment. A more general approach will allow tracking e.g. of living habits and notification of important events like mail arrival.

The paper discusses a power management of devices, wireless and wired communication and data storage on embedded 8-bit microcontroller manufactured by Microchip. A simple analysis of price, performance and memory requirements is done. The practical implementation of the prototype suffered somewhat from lack of lab space and other limitations of the home environment, where the device was assembled.

The achieved results suggest the possibility of such realization. However, the realization has included major changes compared to the originally envisioned device, in order to keep within the limits brought on by the home environment.

The prototype has some design flaws and is a subject to further revision. Nevertheless, the basic idea and solution have proved to be feasible: A modern 8-bit MCU has the capacity to run the intended kind of monitoring and logging system. However the performance is slightly less than hoped for and the experienced memory limitations may suggest a migration to a 16-bit architecture.

I avhandlingen diskuteras skapandet och genomförandet av ett övervaknings- och anmälningssystem för hemmabruk. Ämnet kan ha intresse under den varma svenska sommaren och långa vinter nätter, då väddret sätter krukväxter i hemmet på prov. En livlig fantasi, tillsammans med en livslång erfarenhet av författaren förutspår en total förödelse av gröna växter i lägenheten under sådana förhållanden. En mer generell inriktning av systemet kommer att möjliggöra t.ex. spårning av levnadsvanor och anmälan av viktiga händelser som leverans av post.

Avhandlingen diskuterar möjligheter att implementera en sån enhet, trådlös och trådbunden kommunikation och datalagring i inbyggda 8-bitars mikrokontroller av fabrikatet Microchip. En enkel analys görs av pris, prestanda och krav. Den praktiska realiseringen av prototyren har lidit något av brist på tid och andra begränsningar i hemmiljön, där anordningen tillverkats.
De uppnådda resultaten visar att ett system av den föreslagna typen är möjlig att förverkliga. Emellertid har ett antal förändringar jämfört med den ursprungliga designen varit nödvändiga pga. begränsningar i hemmiljön, där ju tillverkningen skett.

Prototypen har några konstruktionssfel och är föremål för ytterligare revidering. Ändå har den grundläggande idén och lösningen visat sig vara möjliga: En modern 8-bit MCU har kapacitet att köra ett avsett slag av övervaknings- och loggningssystem. Emellertid är slutresultatet inte riktigt det man kunnat hoppas på, och de upplevda minnesbegränsningarna tyder på att en migrering till en 16-bitars arkitektur vore lämplig.
Preface.

I want to thank my mentor Mikael Ekström for his patience and many known and unknown people around me for their will to listen about new and fantastic things I discovered during the project. And racoons, for their strong will to live and fantastic ability to newer give up, especially in situations including sweets.
Table of Contents

Abstract .......................................................................................................................... 2
Preface ............................................................................................................................. 4
Background and related works ....................................................................................... 7
Problem formulation (idea) ........................................................................................... 8
Solution (implementation) ............................................................................................. 11
Main board ..................................................................................................................... 11
  Main MCU minimum requirements ............................................................................ 12
Functional board ........................................................................................................... 15
  Functional board minimum requirements .................................................................. 16
PCB placement and considerations .............................................................................. 18
Workflow and faults ...................................................................................................... 19
Board power .................................................................................................................. 21
  General description ................................................................................................... 21
  Power consumption ................................................................................................... 21
  System voltage ........................................................................................................... 22
Main board power source ............................................................................................ 22
  Power over Ethernet ................................................................................................ 22
  802.3at overview ..................................................................................................... 23
  Main power revised .................................................................................................. 25
Power switch ................................................................................................................ 25
Input rectifier ............................................................................................................... 26
  Choosing a main power step-down: MAX5035A (Maxim) ......................................... 27
Reserve power circuit ................................................................................................... 30
  Switching to reserve power in case of disaster .......................................................... 37
Data storage .................................................................................................................. 43
  General description ................................................................................................ 43
  Secure Digital Card overview .................................................................................. 43
  Microchip MDD File System Interface Library ........................................................ 45
Wireless module .......................................................................................................... 49
  Selecting a wireless module .................................................................................... 49
  Wireless transceiver Microchip MRF24J40MA......................................................... 50
  MRF24J40 ................................................................................................................ 50
  MRF24J40MA ............................................................................................................ 50

5
Wireless framework (MiWi P2P) ................................................................. 51
Device types ............................................................................................. 51
Network structure .................................................................................... 51
Network Addressing ................................................................................ 52
Program interfaces .................................................................................. 53
Ethernet module ....................................................................................... 56
Ethernet on embedded devices ................................................................. 56
SPI base controller ENC28J60 (Microchip) .............................................. 57
External components and PCB considerations .......................................... 58
Microchip TCP/IP stack ........................................................................... 59
Stack architecture .................................................................................... 59
MAC address chips .................................................................................. 65
Making a web page ................................................................................... 67
CSS frameworks ...................................................................................... 67
YAML - Yet Another Multicolumn Layout CSS framework ...................... 68
Folder structure ..................................................................................... 69
Components ............................................................................................ 69
Sending an SMS: SMS Gateway provider .............................................. 70
Summary and conclusions ..................................................................... 71
References ............................................................................................... 72
Bibliography ........................................................................................... 72
Appendix .................................................................................................. 74
Bill of materials (BOM) for the Main board ........................................... 75
Bill of materials (BOM) for the Functional board .................................... 77
Background and related works

There is actually nothing new in the system. Similar systems have been and will continue to be designed by enthusiasts and companies all over the world. Unfortunately, they are often expensive or sold only as part of a bigger solution. The selling company may insist on installation by their technicians and use with their products. Anyway, the manufacturing costs are high due to small quantities manufactured. Another obstacle may simply be lack of outlets for this kind of products in the consumer market.

Similar systems are often called Smart Home, GSM alarm, or even Josh. Some important aspects that a reader must realize about the system are:

- It is not an alarm. Yes, there is a possibility to send a SMS message when an event occurs. But, the communication is not fault proof since it depends on the presence of an up and running Ethernet connection. The communication will be lost in case the connection dies. Another aspect is the wireless communication between the units. The communication can be encrypted, but there is no reserve channel for noisy environments.
- It is not a so called smart solution, although certain small interactions can be added. The solution tends to be easy and inexpensive. A smart solution would require a higher degree of programmability and environmental interaction.
- One-way communication. The idea is a single way communication from a functional board to the main board. The main board is in fact capable of transmitting data to the functional board. However, successful transmission depends on the state of the receiving functional board. Hence, the nature of implementation is single way, from functional board to main board. In other words, the system can be used to turn on the light or something like that. But the signal will be received only at the moment a functional board goes live from sleep. In some cases it may take some time.

So, what is it then? The answer is: This small system provides an easy way to track your habits, notify about events at home, and a way to do something useful once in a while.
Problem formulation (idea).

The idea was to create a system of a main unit, connected to the SMS Gateway Service provider via TCP/IP (internet) on one side, and a set of portable units connected wireless by the main board. The hardware configuration is show at the Figure 4 with the main board on the left side and portable units at the right. The main board has an Ethernet module for communication with external environment and a memory card to store a configuration and logs. Communication between modules is done wireless using an IEEE 802.15.4 protocol.

Main board should provide a user interface as an html-page accessed outside the network. The page should be capable of displaying the current state of portable units.

The logging function is supposed to write occurring events to the memory card on board.

The basic configuration of main unit shall be saved on the Secure Digital card and be editable via Ethernet.

Basically, there are following cases of usage:

- **Configuration:**
  The user can either remotely use the web based interface, or use direct access to the configuration files on the SecureDigital card locally with a PC. The configuration files are used to set the alarms, logging level and restart or shut down the system.

  The functional board has a limited configuration possibility which is not saved on the system and must be retransmitted on every power down; the configuration sets unused ports and interrupts.

- **Incoming event:**
  The incoming event is detected by an interrupt or time base by the functional board. The functional board sends the occurred event to the main board as soon as possible. There is absolutely no knowledge about the structure and meaning of the event at the functional board. A report is sent on a port basis like: Board with address X, PORTA1 is high or INTO has occurred.
The main board receives or requests the incoming transmission and decodes the message using the main board configuration files. After the decoding, the packet is more intelligible to humans. The board X, PORTA1 is now a DOOR_BOX, DOOR_BELL. The next step is to save the decoded event to the log file on the memory card. Once the log is saved, the system determines whether the user requires a notification for the event, and then uses the Ethernet module to connect to the SMS Gateway to send a message. The event is now handled, the system returns to the main loop, and is ready to accept a new event.

- Access the occurred events.
  The logged events can be viewed and analysed. The logs can be accessed through the browser or by direct access to the memory card via PC. The log file uses a CVS-like format and can be loaded into various programs, including Microsoft Visio and Excel in order to add data to graphic applications.

For remote access the embedded web server can also show a log as a web page.
Idea.
A system to monitor and log real-live events

The system consists of a main unit and one or more functional units, connected wirelessly.

The main unit should have an Ethernet connection and act as a web server.

Figure 4: System hardware overview.
Solution (implementation).

As the idea suggests, the board has a number of different components. One thing they have in common is a SPI interface. Brilliant! As the boards have a wireless connection only, they will be discussed separated.

Schematics and PCB design was done in Eagle CAD by CadSoft Software. CadSoft offers a free edition with limited possibilities. The limits of the free copy were reached within a week, which made it necessary to purchase a non-profit licence in order to continue.

The non-profit license has the same features as the Eagle Standard edition. PCB size is limited to 10x16 cm. Schematics allow to use different sheets. The key concept is: One schematic file is one board. A net is connected to another net with names and labels. Nets with the same name are connected; even they are placed at the different sheets.

![Figure 5: Nets and labels.](image)

Figure 5 Illustrate two ways of labelling nets. The OSC1 pin is labelled using an XREF according the format: %F%N/%S.%C%R.
Where %N is a name of the net, %S is a next sheet number and %C and %R are column and row coordinated on the next sheet.

Since version 5.11.0 the Eagle has incorporated a DesignLink interface to Farnell. DesignLink adds additional attributes to components and possibility to add the whole project to Farnell shopping cart.

Because of the size limits of the board and MCU size, SMD components were used. The 1206 and bigger case was preferred. Some components were missing in the standard library and were created.

Main board.

The main board modules are shown in Figure 6. The modules are discussed one by one later in the paper. In order to work, for some modules additional components are required like MAC address chip and power switch. Those are not shown.
Figure 6: Main board modules.

The working conditions of the modules don’t suggest a simultaneous access and modules are used one after another in a manner called cooperative multitasking.

**Main MCU minimum requirements.**
The main unit on the board is an 8-bit MCU. The MCU should have at least one SPI interface. Hardware SPI is preferred. The SPI frequency for the fastest modules is 8MHz and affects the core frequency. The downside of framework usage is that the clock frequency is hardcoded and the program must either switch to another clock source or use the same frequency for all modules. This is often done by a blocking counter.

Microchip MCUs has a few running modes, including an internal oscillator at 8MHz and the use of an external crystal. A usual crystal value of 25MHz with a postscaler of 4 will produce a reasonable frequency.

Despite the fact, that a single SPI is enough to access the modules, a second SPI will provide a better component placement and wiring on the board.

Because of the logging feature of the system, a hardware external or internal real-time clock is a good solution. Embedded hardware module often uses TIMER1 timer and it is uncommon to have a single timer on MCU.

**Memory use.**
As the system uses a few frameworks, the memory can be an issue.

<table>
<thead>
<tr>
<th>Module</th>
<th>Program memory</th>
<th>Data memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory Disk Drive (MDDFS)</td>
<td>34.950</td>
<td>2.215</td>
</tr>
<tr>
<td>Ethernet framework</td>
<td>10.686</td>
<td>156</td>
</tr>
<tr>
<td>Wireless stack (MiWi)</td>
<td>6.836</td>
<td>171</td>
</tr>
<tr>
<td><strong>Total:</strong></td>
<td><strong>52.472</strong></td>
<td><strong>2.542</strong></td>
</tr>
</tbody>
</table>

*Table 1: Main board memory requirements (very appr.).*
An important notice: that memory is required to include libraries into the project. The use of a library will take additional space. Another thing is that every parts of the library, like formatting of the memory card, has its own overhead. Compiler optimizations (for a full version) can remove the unused code bits, but a good idea is to look for more.

The 8-bit MCU has a common memory from 1KB to 64KB or 128KB. The 64KB memory may not be enough and a 128KB has a good margin, and the choice clear.

**Port use.**
Except a memory use, a pin use is a question. Some modules, except the common SPI bus pins and addition CS for every module, requires additional control pins.

<table>
<thead>
<tr>
<th>Module</th>
<th>SPI communication</th>
<th>Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common SPI</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Memory Disk Drive (MDDFS)</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Ethernet framework</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Wireless stack (MiWi)</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Power source</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>MAC chip</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Main and RTCC oscillators</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>Reset and ICSP</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td><strong>Total:</strong></td>
<td><strong>7</strong></td>
<td><strong>16</strong></td>
</tr>
</tbody>
</table>

*Table 2: Main board port usage.*

The simple calculation indicates that a MCU requires at least 23 I/O pins. However, some pins are bound to a peripheral like interrupts and cannot be moved freely. In case of a second SPI module, additional pins are used.

Most devices are available in 6 to 100 pins. The 100 pins devices are usually in a very small TQFP package (is discussed in a fail section) and are hard to solder at home. Lower pin counts are 28, 44 and 64 pins. A 28 pin MCU has been revised and found hard to use due the pin placement in hardware. Therefore, a 44 pin MCU is a much better choice with more freedom to place devices around. Package has a larger TQFP package too.

**Selecting a MCU.**
Microchip web page has a product selector section. Following information has been inserted into the selector.

- 8-bit architecture.
- Flash: >128KB.
- RAM: >2KB.
- Package: 44 pins
- SPI: 2+
- Real Time Clock: H/W RTCC
During a product selection, a single consideration was done. The data memory use was set to 2KB. The actual requirement can be reduced due to the compiler and by disabling of unused components inside the module.

The search result produced only two options: PIC18F47J13 and PIC18F47J53 (Microchip). Those are very similar, except that the second has an embedded USB peripheral.

The search results give a good indication that an 8-bit MCU may not be enough in a future. In fact, a migration to a 16 or 32 bit architecture may be required on later products.

**Main MCU: PIC18F47J13.**

The selected MCU is a part of a family with four microcontrollers. Each of them has a low power version, making a total amount of MCUs to eight. MCUs are in 28 or 44 pin packages with 64KB or 128KB of program memory and a data memory of 3.760 bytes. The available data memory actually is larger than required and leaves app. 1KB free.

The 44-pin TQFP version of MCU has a feature called Peripheral Pin Select that allows hardware peripheral pins to be moved across redefined pins in software, including SPI. The actual MCU has 25 remappable pins.

Additional features include a flexible oscillator structure, a four external interrupts, a Charge Time Measurement Unit (CTMU).

<table>
<thead>
<tr>
<th>Amount</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 – 9</td>
<td>4.40€ (50.83 SEK)</td>
</tr>
<tr>
<td>10 – 99</td>
<td>3.31€ (38.13 SEK)</td>
</tr>
<tr>
<td>100 +</td>
<td>2.83€ (32.64 SEK)</td>
</tr>
</tbody>
</table>

Table 3: Price of a PIC18F47J13 MCU.

Since the number of free pins available, the HS oscillator has been added to the design (the original plan was to use an embedded crystal) and three pins were connected to a pin-header for testing purposes. All other unused ports were set as outputs and connected to the ground. The primary reason for that is a lack of space on the PCB.
The main board has been produced as a prototype on a 10x16cm Euro board.

**Functional board.**

The functional boards were thought as portable, low-power small units. The board has a single wireless module connected to it. Remaining ports are used to read the sensor values. The core clock is, for the most, disabled or operating at a low frequency. The power consumption has a higher priority since the board has battery power only.

![Functional board](image)

*Figure 7: Main board prototype.*

*Figure 8: Functional board modules (MAC not shown).*
Functional board minimum requirements.

The memory use of board is low. Additional space is taken by the software configuration wizard and is unknown.

<table>
<thead>
<tr>
<th>Module</th>
<th>Program memory</th>
<th>Data memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration wizard</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Wireless stack (MiWi)</td>
<td>6.836</td>
<td>171</td>
</tr>
<tr>
<td><strong>Total:</strong></td>
<td><strong>6.836</strong></td>
<td><strong>171</strong></td>
</tr>
</tbody>
</table>

Table 4: Functional board memory usage.

The suggested MCU should have 8KB or better 16KB of flash memory and 500+ bytes of data memory.

The port requirements are low as well, and depend on the number of connected sensors.

<table>
<thead>
<tr>
<th>Module</th>
<th>SPI communication</th>
<th>Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common SPI</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Wireless stack (MiWi)</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Power source</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>MAC chip</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Reset and ICSP</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>Other peripheral (sensors)</td>
<td></td>
<td>6 (typ)</td>
</tr>
<tr>
<td><strong>Total:</strong></td>
<td><strong>5</strong></td>
<td><strong>9+</strong></td>
</tr>
</tbody>
</table>

Table 5: Functional board port requirements.

The hardware SPI is not needed and can be emulated in software. An RTCC module is a plus for some applications, but is optional.

Both these requirements make it possible to use virtually any MCU.

Selecting a MCU.

Following information has been inserted into the selector.

- 8-bit architecture.
- Flash: 8KB – 64KB.
- RAM: >0.5KB.
- Package: 14 - 28 pins
- Low power: XLP

The search result produced a few pages with a cheapest match for a PIC16F1825 MCU.

PIC16F1825 come in a 28-pin PDIP, SOIC, SSOP packages with 25 I/O ports.

<table>
<thead>
<tr>
<th>Amount</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 – 25</td>
<td>2.03€ (23.40 SEK)</td>
</tr>
<tr>
<td>26 – 99</td>
<td>1.85€ (21.41 SEK)</td>
</tr>
<tr>
<td>100 +</td>
<td>1.68€ (19.41 SEK)</td>
</tr>
</tbody>
</table>
MCU is classified as a new product and the price may change with time and is taken to show the lowest price.

The actual MCU used in the prototype is a small brother of the main MCU: PIC18F26J13. MCU has 28-pin SOIC package and 64KB program memory with all family features. The price is almost doubled and is around 4€.

![Functional board breakout.](image)

Since the functional board has lower requirements it was done as breakout modules on a project board. The final PCB was still designed as reference in two versions. The functional board fits the half of the Euro card (10x16 cm) and two options were considered: 5x16cm and 10x8 cm. The longer, 5x16cm, option looks better, but the enclosure is hard to find.
PCB placement and considerations.

The main PCB is done using a Star-ground configuration. Each module has its own ground polygon which is connected to other polygons at a certain points.

The board has noisy DC-DC converters that are placed at top left corner. The wireless transceiver is placed at another side of the board to avoid interference. The Ethernet module due its high power consumption is placed close to the power source.
Workflow and faults.

The project has a lot of problems and faults (see Figure 11).

The original idea was to use a PIC18F97J60 (Microchip, 2009) controller with embedded Ethernet module. The design was a very compact and without an external transceiver IC.

The Ethernet family of Microchip MCU includes nine different microcontrollers in 64/800/100 pin TQFP package. The program memory varies from 64K to 128K and data memory is 3808B for every device. The PIC18F97J60 has a 128K memory, 64-pin package with 39 I/O pins and one hardware SPI module. A larger, 100 pin, variant have an External Memory Bus module and the MCU memory can be extended by parallel external EEPROM, Flash or SRAM memory.

The problem came from an unexpected direction: the footprint of MCU was too large. A quick solution was found and ordered: a TQFP64 Adapter and the PCB layout were completely remade.
Upon the receiving of the adapter (app. 2 weeks) the second surprise has come. The received adapter was too big as well. By that time, the author has realized the drollery and started looking for answers. The answer was rather simple: there were two TQFP64 packages with 0.5 and 0.8 pitch distance. The soldering of 0.5 pitch wasn’t possible and the other solution with external transceiver was used (that required the removal of an SPI Real-Time Clock).

Around that time, the Power over Ethernet (PoE) was discovered as an option to power the main board. The results has shown some negative aspects and the PoE was removed from the project. A short description is presented at the main power chapter.

To avoid more size related supersizes, a one-in-a-time approach was made. The solution included a pin out board for every module and a simple testing. That didn’t produce any unexpected results and a final assembly was made.

The final assembly, however, gave an interesting problem. The MAX866 DC-DC converter didn’t match the desired place on the board, mostly, because of many external components required to operate. A quick replacement in form of Sipex 6641B was produced.

The last problem came from the power switch circuit MAX804R. The maximum output current of the switch was only 100mA, instead of 300mA required to run the entire system. The problem is: There is not so easy to find a low-voltage power switch. The MAX804 was the only available search result that could be delivered in time.

The replacement of IC on a ready board is not possible. A complete component removal is required and because of the component cost and time needed to reassemble the board the project was aborted.

Because of that, the programming part couldn’t be done at the final prototype.
Board power.

General description.

The power for the board uses two different circuits to power main and function boards. The key difference is that the main board requires a reserve power source to properly shut down the circuit and a function board is always powered by battery source.

When the main power down occurs the following modules are dependent:

- Open files on SD card must be closed and the power down notification written or the file system may be damaged.
- The radio module must be put to sleep or disabled. The module includes a retransmit on failure capability and may lead to excessive traffic.
- Ethernet module to finish operations and send power is down command. An interrupted communication can’t actually damage the module, however many open connections to remote server can exceed server limits and lead to unavailability to other users.
- MCU save active state if needed.
- Real-time clock continue to run.

Those actions are taking place with the main power already down and can’t be interrupted. The power down sequence doesn’t take fixed amount of time and may vary depending on active module.

Power consumption.

To ensure that current limits are met, a good idea is to look at system power consumption. In the actual system, the power can be used more efficient by disabling unused modules or by using sleep mode. The actual current consumption may vary, depending on oscillator frequency and power mode.

<table>
<thead>
<tr>
<th>Module</th>
<th>Run (active)</th>
<th>Reduced mode (idle, standby)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCU</td>
<td>1.9 mA, PRI_RUN, 4MHz</td>
<td>53 – 150 nA (deep sleep),</td>
</tr>
<tr>
<td>SD card</td>
<td>50-100 mA</td>
<td>15 mA</td>
</tr>
<tr>
<td>MRF radio module</td>
<td>23 mA</td>
<td>2 uA</td>
</tr>
<tr>
<td>Ethernet module</td>
<td>120 - 180 mA</td>
<td>2 mA</td>
</tr>
<tr>
<td>EEPROM</td>
<td>5 mA</td>
<td>1 mA</td>
</tr>
</tbody>
</table>

Table 7: Power consumption.
As can be seen at the table above, the main system may require about 300mA during runtime for functional board and 30mA for functional board. The actual design can produce up to 1A output.

**System voltage.**

Another important question is a system voltage. Different voltages can require level shift.

<table>
<thead>
<tr>
<th>Module</th>
<th>Voltage</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCU</td>
<td>2.15 to 3.6V</td>
</tr>
<tr>
<td>SD card</td>
<td>2.0 to 3.6V</td>
</tr>
<tr>
<td>MRF radio module</td>
<td>3.3V (2.4 to 3.6V)</td>
</tr>
<tr>
<td>Ethernet module</td>
<td>3.3V (3.1 to 3.6V)</td>
</tr>
<tr>
<td>EEPROM</td>
<td>1.8 to 5.5V</td>
</tr>
</tbody>
</table>

*Table 8: Operational voltage.*

All components can perform from 3.3V supply.

**Main board power source.**

**Power over Ethernet.**

The idea of using Power over Ethernet (PoE) came up quite early as an idea. It fits well, completely matches the requirements and most of all, it provides a possibility to remove power wire off the board. Not to mention cheaper cabling, independency of local variations of power standards and plugs. PoE solution deploys everywhere where Ethernet is.

Possible application includes VOIP, network devices like webcams, phones, routers and project like this one. Provided power of up to 15.40W is more than enough to power up a small PC.

PoE in this project was discovered from the PIC18F97J60 MCU manual as notice downside the figure 18-2: EXTERNAL COMPONENTS REQUIRED FOR ETHERNET OPERATION inside the PIC18F97J60 datasheet. A quick search gave a very optimistic, easy to implement solution.

**Power injection into spare wires.**

Power injection is probably the easiest way to implement PoE device. The Unshielded Twisted Pair (UTP) wire has eight threads, twisted two-by-two and leaving four threads free. These threads can be used to inject power into one side and pick it from another. Unfortunately, there are many applications and hobbyists thinking in the same pattern.
Spare wires can, for example, be used for signal transmission by older phones, by network providers to save costs or by gigabit Ethernet.

Pairing the real PoE device with such equipment won’t damage it, but other devices, including powering device, may stop working. As a consequence of such mistake, power may get in contact with human body in form of shock or simply drain the source in case of too high consumption.

Despite that, the power injection is the easiest and fastest way to adapt existing device to PoE.

In order to regulate it, the IEEE association comes up with an 802.3af/at standard.

**The 802.3af/at standard.**
The specification was released in 2003(802.3af) and updated 2009(802.3at). Rather than using spare wires to transfer the power, the data-wires are used with phantom technique. Power is added using the signal transformers center taps and taken the same way. The resulting coupling is very robust, fault-free and complex, including such things like: polarity independency, short-circuit protection and will work with not PoE enabled devices.

At this point, PoE has become a complete protocol, including detection, classification and operation states. Special class of devices called PoE powered device controllers are usually used to implement it.

**Mixed mode.**
The mixed mode allows combining previous two techniques in 802.3af devices will still work, however without all protection and doesn’t guarantee the safety of devices. Both techniques are covered next.

**802.3at overview.**
The PoE was taken to provide 10BASE-T, 100BASE-TX, or 1000BASE-T communication over the same interface with the power to process the data. Unlike a regulated or unregulated power source, DTE (Data Terminal Equipment) has unique features. A regular power source, is always providing power to the target, the PDE doesn’t. PDE can provide power on-demand, based on powered device load requirements. Further, it can allocate power while it is still connected or completely disconnect it if needed. Powered equipment, on the other side, can request required power if needed.

PoE has same simplicity as usual Ethernet equipment from the user point and can be used transparent together with other equipment.
The description of PoE can’t be complete without mentioning the device types. Devices are divided into power sourcing equipment (PSE), powered devices (PD) and power interface (PI).

PSE main function is to search for PD and supply power to it. When the PD is disconnected, the power is removed from the endpoint, providing safety. PSE won’t provide any power to PD if its link is unable to provide that power.

PD is a device that requests and consumes power from PSE, but it can also have its own power source and not take any power from PI at all. Powered devices are polarity insensitive and should withstand voltages from 0-57V and requested power without taking damage. Unlike the PSE, PD should not source power to power interface.

Both PSE and PD or connected equipment should provide isolation between all conductors, including frame ground and withstand voltage up to 1500V RMS for 60s without breakdown.

![Figure 15: TPS2376 typical application circuit with ORing.](image)

The TPS2376 (Texas Instruments, 2008) is a Powered Device controller and shows a typical application circuit including a rectifier and data lines. The ORing can be implemented in three options.

1. To the output of DC-DC converter, an easy way to add PoE to existing devices.
2. At the PD output.
3. Before the PD circuit.

Each option has its own advantages and disadvantages. More information is available at the TPS2376 Datasheet, page 14.
The PoE design came after the project start and wasn’t included into the schedule. Unfortunately, the components have a long delivery time and cost and there were absolutely no place for them on the PCB. A quick search on consumer market (Prisjakt.se) revealed only two home products with PoE support. None of them looked nice and promising and the PoE idea was removed from the project.

**Main power revised.**

The main board power is partially affected by earlier power over Ethernet idea, which was dropped. When that happened, the decision has to be made, wherever keep the idea of variable, up to 100V input voltages for main power, or use the lower voltages of widely used in customer electronics adapters, around 5V DC.

Anyway, the main input voltage must be scaled down to the 3.3V DC system voltage, described in this chapter. The reserve battery voltage, on the other side must be used wisely and efficient. In order to archive the longer live, the board must operate from voltages below the rated battery voltage, in range of 1,5V to 9V DC.

The current decision is one of the key points of how the final power board will look like, that affects even a switchover circuit.

### Lower voltages:

**Advantages:**
- Can scale down using a LDO or LDO cascade from up to 24V.
- Cheap.
- Small footprint.
- Efficient.

**Disadvantages:**
- Power dissipation and heat.
- Limited input voltages.

### Higher voltages:

**Advantages:**
- More universal.
- Future use with PoE.

**Disadvantages:**
- Price.
- Footprint.

The solution was to leave the earlier high voltage circuit, despite the benefits of lower power solution and use a DC/DC step-down converter.

**Power switch**

There are two major solutions for power switch.
The actual power configuration uses two different DC-DC converters to scale the input voltage to 3.3V system voltage and a power switch to select either of them.

Another option was to use a high voltage switch and only one DC-DC converter with wider input voltages.

In that configuration an unregulated voltage is passed to a voltage switch for selection and to a single DC-DC converter. This variant is easier to implement, but components were rather hard to find and get.

**Input rectifier.**
To ensure the right polarity of power source, a 4 diode bridge rectifier is implemented before the converter on 1N4004 diodes. 1N4004 can handle up to 400V, 1A with forward voltage of 1.1V. In combination with input bypass capacitor of MAX5035 will produce unregulated DC voltage. The forward voltage is rather high and may result in power loss and voltage drop. A diode with lower voltage is preferred, like one in converter circuit.
Choosing a main power step-down: MAX5035A (Maxim).

The converter selection was rather simple. There were two major competitors in IC selection: MAXIM or Texas Instruments. Unfortunately, TI packages were rather small and hard for home soldering. Ready to go solutions were dropped because of price or size.

Along the possible solutions, MAX5035 was chosen.

MAX5035 is a step-down DC/DC controller, operation at 125 kHz with pulse-skipping mode for lower quiescent current and higher performance at higher loads. The circuit consumes only 270uA under no load conditions and 10uA in shutdown.

Features:

- Wide 7.5 to 76V input voltage range, with maximum input voltage on \( V_{in} \) of 80V.
- Fixed (3.3V, 5V, 12V) or adjustable output packages.
- 1A output current.
- Efficiency up to 94%.
- 8-pin SO or DIP packages.

![Figure 18: Typical operating circuit (Maxim, 2010).](image-url)
**Undervoltage lockout and shutdown mode.**

Circuit can be shut down, minimizing the supply current to 10uA using the ON/OFF pin, driven low. The ON/OFF voltage must exceed 1.69V before the circuit is operational. Same pin is used to program the Undervoltage lockout (UVLO) using a simple resistor divider. Under certain conditions, the UVLO can be omitted. In such case, the input voltage \( V_{in} \) is used as reference and circuit will be started, when \( V_{in} \) is greater than 5.2V.

UVLO is calculated using the formula:

\[
V_{UVLO(TH)} = \left(1 + \frac{R1}{R2}\right) \times 1.85V
\]

**Figure 19: UVLO formula.**

The recommended value for \( R2 \) is less than 1MOhm and recommended voltage for 3.3V output is 6.5V.

**Component selection.**

**Undervoltage lockout value.**
Based on \( V_{uvlo} \) and datasheet recommendations, \( R2 \) is chosen to 1MOhm and \( R1 \) to 390kOhm. These values provide the \( V_{uvlo} = 7.21V \), a fully accepted value.

**Inductor.**
The inductor choice is guided by the voltage difference between input and output voltages, output current and operational frequency. The operational frequency is 125 kHz and the peak switch current limit (\( I_{lim} \)) is 1.80A (typ) and 2.5A (max). The inductor maximum saturation current rating must be at least equal or greater than \( I_{lim} \). Inductors with low DC resistance are preferred for higher efficiency.

The actual used inductor is a 100uH, LQH6PPN101M43L, produced by Murata with maximum DC current of 0.8A, packed in surface mount 2424 case. The current rating of inductor is lower than desired, but the converter will still work because the system current consumption is below that value.

The inductor that should be used is 100uH, WUERTH ELEKTRONIK 1635948 inductor or VISHAY DALE IHLP4040DZER101M11 inductor with the price of 2.5€/piece (25 SEK). The actual replacement LQH6PPN101M43L has a price of 0.35€/piece (3 SEK).

**Rectifier diode.**
The converter requires an external diode rectifier, placed close to the converter with short traces. Diode should have continuous current rating greater than expected output current.
and voltage rating greater than $V_{in}$. A low forward-voltage Schottky diode is a good option, in order to keep temperature down, suggested $V_{fb}$ is lower than 0.45V.

A product search produced a single hit on VISHAY VS-30BQ100TRPBF, SMD case SMC for 0.98€/p. The package is hard to place and route on PCB and a less precise search produced following options:

- Datasheet recommended VISHAY 50SQ100, 100V, 5A, $V_f = 660mV$ in DO-204AR package for 3.03€ (29.33 SEK). Other options are not produced or available.
- ON SEMICONDUCTOR MBR3100G, 100V, 3A, $V_f = 790mV$ in 267-03 package for 0.75€ (6.94 SEK).

The last one was chosen as best value for money alternative. With lower input voltages other options are possible like 1N5817, 20/30/40V, 1A, with considerably lower price.

**Capacitors.**

Two 0.1μF ceramic capacitors are connected to circuit. A boost capacitor, 16V, connected between BST and LX pins and internal regulator output capacitor, connected to $V_d$, 16V (9.0V max), bypassed to ground.

**Input bypass capacitor.**

The high switching frequency of MAX5035 requires lower capacitor values. Low-ESR aluminium electrolytic capacitors with high ripple-current capability can be used. For better performance at cold temperatures, a 1μF ceramic capacitor is used in parallel. In case of size-optimized applications, a low-ESR, ceramic, multilayer chip capacitors are preferred.

Capacitor should be capable to handle current ripple. Suggested value for input bypass capacitor is 68μF, 100V.

**Output capacitor filter.**

The output ripple requires output capacitor, low-ESR tantalum or aluminium electrolytic capacitors are suggested or a combination of low-ESR tantalum and ceramic capacitors for better transient load and ripple/noise performance.

Suggested value is 68μF, 10V.

**PCB layout considerations.**

The anode of rectifier diode, the input bypass capacitor lead, output filter capacitor lead should be connected together in “star ground” configuration on a ground plane to minimize the ground noise. Diode, capacitors should be placed close to device.
Reserve power circuit.

The reserve power circuit is typically a battery with some kind of regulating IC. The different power consumption of main and functional boards makes the choice more difficult. A great option would be to use the same circuit for both or to use different versions of pin-compatible IC. Same consideration can be applied to reserve power as for main power. Either use bigger, 9V batteries or cascade with LDO, or for smaller CD2032 or AA size batteries and more complex DC/DC buck-boost adapter.

Batteries differ in size, capacity and drain. The most used and compact battery is a CR3020 cell battery. It is surface mounted, with an output voltage of 3.0V and 225 mAH output and weight of 3g. Unfortunately, low-drain.

AAA alkaline or lithium cell 1.5V batteries have a higher drain capacity and more than 1200 mAH output. The size and weight of 12g makes them not suitable for PCB soldering.

The first choice was a MAXIM MAX856 from MAX856/859 family of CMOS based DC-DC switching regulators for low voltages with output current of 500mA. The circuit was soldered and tested on an external board.

A last-minute surprise came from unexpected direction – it was too large for main board in its current state. Another, smaller replacement was found, Sipex SP6641B family.

Both IC has advantages. MAX856 is more expensive that SP6641 and has more external components, but its efficiency and structure allows the use of cheaper components like general purpose electrolytic capacitors and small, low-cost inductors. Sipex is cheaper with fewer components, but requires bigger and better inductor and large ceramic capacitors, that is a major contribution to the total circuit price. The output voltage ripple is high and the use of 3.0V LDO or LC filter is suggested by the datasheet.

<table>
<thead>
<tr>
<th></th>
<th>MAX856</th>
<th>SP6641B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output current</td>
<td>500mA</td>
<td>500mA</td>
</tr>
<tr>
<td>Output voltage ripple</td>
<td>Fine</td>
<td>Worst</td>
</tr>
<tr>
<td>Input voltage</td>
<td>0.8V to 6.0V</td>
<td>0.9V to 4.5V</td>
</tr>
<tr>
<td>Output voltage</td>
<td>3/5V pin selected or adjustable</td>
<td>Fixed</td>
</tr>
<tr>
<td>Components</td>
<td>Low value, cheap capacitors and inductor</td>
<td>More expensive ceramic capacitors and inductor.</td>
</tr>
<tr>
<td>Number of components</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>Number of pins</td>
<td>8</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 9: MAX856 and SP6641B comparison.

Maxim has a wider input voltage and a set of useful features like programmable low battery output and 1.25V voltage reference output and shutdown pin. Because of the work done on the subject, a short description is following.
**Reverse battery protection.**
The main power source is protected from wrong polarity by a full-bridge diode rectifier. The downside of that is a voltage drop across a diode, which decreases already low input voltage.

Another solution is to use a MOSFET protection.

![Figure 20: Reverse battery protection with P- MOSFET.](image)

The circuit works because of a body diode. The positive voltage is applied to the diode and it starts to conduct. The voltage appears at the drain, except the diode drop voltage. As the load receives the voltage the diode is shorted.

A chosen MOSFET needs low threshold voltage and pass at least 0.5A as well as low resistance.

NXP PMV65XP,215 P-Channel MOSFET in SOT-23 package was chosen, with $I_d$ of -2.8A, $R_{on}=76m\Omega$ and $V_{gs}=-750mV$. The prices are shown below.

<table>
<thead>
<tr>
<th>Amount</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 – 24</td>
<td>0.20€ (2.36 SEK)</td>
</tr>
<tr>
<td>25 – 99</td>
<td>0.154€ (1.77 SEK)</td>
</tr>
<tr>
<td>100 – 249</td>
<td>0.128€ (1.48 SEK)</td>
</tr>
</tbody>
</table>

*Table 10: NXP PMV65XP,215 P-Channel MOSFET prices.*

Unfortunately, somehow during the converter switch it disappeared and is not on the current board, but will come back in next revision.
**Dead end: MAX856, DC-DC converter.**

The MAX856 is a part of MAX856-MAX589 family of DC-DC switching converters with pin-selector or adjustable voltage output. Circuits are partly pin-compatible and available in uMAX, SO and CERDIP packages. An internal MOSFET and high switching frequency allows using a cheaper inductor and capacitors.

The 3/5 pin is used to select the output voltage and should be connected to OUT or GND. Any voltage outside that range can damage the circuit. The used configuration is for 3.3V and pin is connected to the OUT. The other control pins are LBO (low-battery output) and shutdown (SHDN) are not dependent on the input voltage and be pulled up to +7V.

The shutdown (SHDN) input can be used to disable the converter. While disabled, the output voltage follows the input voltage, except the voltage drop across the diode. In other words, the circuit is powered.

The price of MAX856CSA+ is shown in table below.

<table>
<thead>
<tr>
<th>Amount</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 – 9</td>
<td>3.52€ (54.79 SEK)</td>
</tr>
<tr>
<td>10 – 99</td>
<td>2.82€ (35.25 SEK)</td>
</tr>
<tr>
<td>100+</td>
<td>2.00€ (23.77 SEK)</td>
</tr>
</tbody>
</table>

Table 11: MAX856CSA+ prices.

As seen, the price change is significant. The price of external components can vary, depending on circuit output current and voltage ripple. The large, cheap, electrolyte capacitors or smaller and more expensive ceramic capacitors can be used. The inductor value range is large too.

![Figure 21: Typical operating circuit, MAX856 (Maxim, 1996)](image)

Key features:

- 0.8V to 6.0 input voltage.
- 85% efficiency.
- 25uA quiescent and 1uA shutdown currents.
- Up to 500 kHz switching frequency.
- Programmable low-battery detector.
- 1.25V voltage reference output.
Voltage reference.
A precision, 1.25V, 2% voltage reference is available at pin 3. This output is capable of driving a 250µA and sinking up to 20µA loads and must be bypassed to ground with at least 0.1µF ceramic capacitor.

Low-battery detection.
The circuit can sense the input voltage on LBI pin and using the comparator and internal voltage reference sink the open-drain, LBO output, to ground, when the input falls below the reference voltage. The threshold is set by resistors R3 and R4 in Figure 21 using the formula.

\[ V_{	ext{LBO}} = V_{	ext{ref}} - \left( \frac{R_3}{R_3 + R_4} \right) \]

The low input current to LBI minimizes loading of the input supply. With the value of 100nA, the recommended resistors are in the range of 300 kΩ. Since the LBO is an open-drain output, the 10 kΩ pull-up resistor is required when driving a CMOS circuit.

The converter is operational without the detection circuit. When the battery detection is not used, the LBI should be connected to \( V_{\text{in}} \) and LBO open.

Component selection.

Inductor.
The efficient switching and high switch frequency allows to use a wide range of values in the range from 10uH to 100uH, depending of physical size, output voltage ripple and current. The recommended value is 47uH with incremental saturation current rating of 500mA and low DC resistance.

The used inductor is a MURATA LQH6PPN470M43L, 47uH in 2424 case, with DC current rating of 1.1A. The price is 0.27€ (3.08 SEK)/piece.

Capacitors.
The same logic is applied to capacitor selection. Low-ESR capacitors are recommended, especially tantalum for lower temperatures. A 68uF, 10V tantalum surface-mount capacitor is a good choice, however the electrolytic capacitors is also suitable with some adjustments.

Electrolytic capacitors are rather cheap, with the price of 0.21€ (2.40 SEK) for 10 pieces (MULTICOMP MCESL10V686M6.3X5.2). Tantalum, AVX TAJW686K010R has a price of 0.55€ (5.78 SEK) for 5 pieces and ceramic capacitors are about 2€/piece.

Rectifier diode.
For better performance a Schottky diode with low forward voltage, like 1N5817, is recommended.

The price of 1N5817 diode starts from 0.16€ (2.33 SEK).
PCB layout.
Leads between C1 and C2 should be less than 5mm and IC ground connected direct to the ground plane.
Replacement: Sipex SP6641B (EXAR) DC-DC converter.

SP4441 is another DC-DC boost converter by EXAR, targeting batteries in portable devices in SOT-32, 5-pin package. The circuit is available in 3.3V and 5V fixed voltage options. The output current depends on input voltage. SP6641B have an output up to 500mA and SP6641A around 200mA.

Prices are shown below:

<table>
<thead>
<tr>
<th>Amount</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 – 9</td>
<td>1.38€ (14.40 SEK)</td>
</tr>
<tr>
<td>10 – 99</td>
<td>1.07€ (11.15 SEK)</td>
</tr>
<tr>
<td>100 - 249</td>
<td>0.87€ (9.22 SEK)</td>
</tr>
</tbody>
</table>

Table 12: Sipex SP6641B prices.

The versions of IC are pin-compatible, but schematic requires small changes. For the B version, C3 must be left open and R1 shorted.

Key features:

- 10 uA quiescent current.
- 0.9V to 4.5V input voltage.
- Up to 87% efficiency.
- Shutdown input.

The circuit is rather noisy and a 3.0V LDO or LC filter is suggested at the output.

Component selection.

**Inductor.**

The suggested value of inductor is 10uH, the saturation current of inductor needs to be greater than IC peak current, which is 1.0A (TYP). A low DCR value will increase efficiency.

The used inductor is a BOURNS SDR0604-100ML, 10uH in SMD case, with \(I_{\text{RMS (MAX)}}\) of 1.45A and \(I_{\text{SAT (TYP)}}\) of 2.0A. The price is 0.53€ (6.11 SEK)/piece.
**Capacitors.**
A use of general purpose low-ESR capacitors is not recommended because of major contribution to output ripple. Ceramic capacitors are preferred for lower ESR, another option is a tantalum capacitors. The price of tantalum capacitors with high value is generally lower, than for corresponding ceramic capacitor. Attention should be paid to current ripple parameter.

The value of both capacitors is suggested to 100uF.

The used capacitors are ceramic TAIYO YUDEN, JMK325F107ZM-T SMD in 1210 case with voltage rating of 6.3V and price of 1.55€ (10.29 SEK)/piece.

An alternative is a tantalum VISHAY SPRAGUE, TR3D107K010C0150 in D case with voltage rating of 10V and price of 0.58€ (7.39 SEK)/piece.

**Rectifier diode.**
The chosen rectifier was an SMD Schottky rectifier by FAIRCHILD SEMICONDUCTOR, MBR0520L in SOD-123 package, 20V, 385mV forward voltage and $I_{F(AF)} = 500mA$. The forward current could be larger.

Rectifier is available for 0.198€ (1.48 SEK)/piece.

**PCB layout.**
The components should be placed short and wide, with traces larger than 50mil (1.25mm). Components should be connected in a “star”-ground configuration with filter capacitors as close as possible. IC and leads should be placed far from sensitive inputs and RF parts.
Switching to reserve power in case of disaster.

When the main power goes down, the system must change to reserve power from the battery source. There are several commonly used techniques to achieve that and several things to consider: the power source shouldn’t damage other source with reverse current, switchover must be fast and voltage kept above threshold and with minimum voltage drop.

Despite the main purpose of distributing the load a power switch can include other features like limiting inrush current during start-up, power distribution to sub-systems and short circuit protection.

**Analogue or MCU based approach.**

**Diode ORing.**

Diode ORing is probably the easiest way to combine two sources and one load. The circuit represents a logical OR gate. The diode is blocking the lowest voltage; careful choice of components gives more accurate control over the circuit.

![Diode ORing switch.](image)

Despite the easy implementation, there are some drawbacks, especially for low voltage systems.

**Losses during the insertion into power path:**

- **Voltage drop.** A regular diode has a drop around 1.2V and a fine Schottky rectifier can have as low as 330mV. It seems to be low; the voltage of 3.3V system will sink below the operating voltage.
- **Power loss.** The load current is passed through the leading diode, causing a power loss of \( (\text{voltage drop times load current}) \) and heat in diode. The estimated power low will be

  \[ \text{An average SOD123, SMD, package is able to dissipate up to 400mW.} \]

**Losses during leakages:**
A regular or Schottky (mostly) diode has some losses in off state. Those losses are around a few mW.

**FET ORing.**
For lower power losses a diode can be replaced by a FET MOSFET, as described earlier. The circuit is more difficult to implement.

![FET ORing switch (P-MOS).](image)

Except for complexity, the slower turn-off time and trip voltage point calculation is a problem. The worst-case turn-off time of Vishay Siliconix Si2305CDS is 60ns and 30ns for turn-on. During that time, the circuit can be shorted or unstable.

The power loss can be calculated as (for $V_{GS} = -2.5$ V, $I_D = -3.8$ A, max):

$$P_{loss} = (2.5 \times -3.8) = 0.96W$$

And power dissipation of 0.96W in SOT-23 case.

A special attention should be paid to MOSFET selection: a regular MOSFET may not include the reverse voltage protection. In that case, the input voltage should always stay higher than the output voltage or a large current will flow back to input. There are a group of IC called MOSFET ORing controller to ensure a safe operation of ORing. Configuration is possible for N+1 power sources (look at TPS2412/13).

This solution came at second place with a very small margin.

**Switchover with shutdown pin.**
The next way to achieve switchover is using the SHUTDOWN pins on both systems. MAX5035 and similar IC, for example, have a programmed, active low power fail output
(PFO) pin. When the voltage drops below threshold, the pin is driven low and IC is shut down. A reserve power IC detects voltage drop across the shutdown pin and drives the voltage to desired level. The procedure is repeated backwards in case of voltage return to the main source.

![Figure 25: Switch using SHDN and PFO.](image)

Unfortunately, some protection for reverse currents is required for every IC. Another risky moment occurs when the voltage change is slow and both IC are driving the output.

**MCU switchover.**
A modern MCU has a set of features that can be used to monitor and control the power. One of the MCU used in this project, has a High/Low-Voltage Detect (HLVD) module and three comparators.

HLDV module can monitor high or low voltage on V_{DD} or HLDVIN pin and compare it to internal voltage reference voltage and trigger an interrupt. A trip-point is hardware set using a resistor divider on two resistors.

![Figure 26: MCH switch.](image)

When the interrupt occurs, the software can drive a corresponding shutdown pin of regulator and ensure that timing and voltage limits are met, as well as pause an operation during switchover.
Voltage supervisor with battery backup (MAXIM MAX804R).

A more advanced approach is to use a ready component like MAX804R. The circuit is a voltage supervisor with battery backup for 3.3V systems. The IC is a part of a bigger family with six other supervisors. Circuit is available in SO and DIP packages.

Prices are shown below:

<table>
<thead>
<tr>
<th>Amount</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 – 24</td>
<td>10.68$</td>
</tr>
<tr>
<td>25 – 99</td>
<td>7.12$</td>
</tr>
<tr>
<td>100 +</td>
<td>5.16$</td>
</tr>
</tbody>
</table>

Table 13: MAX804R prices (from MAXIM website).

Features:

- Reset output
- Supply voltage monitor
- Watchdog timer
- Battery-Backup Power Switching
- Low-Battery Warning

A first letter after the circuit number denotes the threshold voltages: 3.075V (T), 2.925V (S), and 2.625V (R). The used IC is an MAX804R, but a MAX804T should be really used.

Figure 27: Typical operation circuit: MAX804 (Maxim, 2005).
The main power is connected to $V_{CC}$ and the output voltage is taken at $V_{OUT}$. A switchover occurs when $V_{CC}$ falls below $V_{SW}$ (2.40V) and $V_{BATT}$. $V_{OUT}$ switches from $V_{CC}$ to $V_{BATT}$ until $V_{CC}$ rises above the reset threshold and $V_{OUT}$ reconnects to $V_{CC}$.

The battery can be replaced while the $V_{CC}$ stays above the $V_{SW}$ without triggering the reset if a ceramic capacitor of 0.1uF is placed at the $V_{BATT}$.

A $V_{CC}$ (source) and $V_{batt}$ are rated up to 6V and $V_{batt}$. The battery voltage can be higher than the main supply input voltage. There are some negative things about the IC as well. The Continuous Input Current is only 100mA for $V_{cc}$ and 18mA for $V_{batt}$. The output current of $V_{OUT}$ is 100mA. These values are below the current requirements of the system and can cause problems in the future and probably replaced with one of Texas Instruments or Maxim load switches or ORing MOSFET controllers.

![Figure 28: Simplified connection diagram for MAX804.](image)

The $V_{cc}$ is connected to main power (MAX5035) output and $V_{batt}$ to battery DC-DC converter (SP6641), both produce regulated 3.3V output. PFI connected to battery and will trigger the PFO when battery needs to be replaced. The allowed voltage on the PFI pin is bound to the maximum input voltage of 6.0V and the maximum input of SP6641 is 4.5V.

**Features.**

**Bi-directional RESET.**
As a voltage supervisor, the IC has a set of features to communicate to MCU. A bi-directional RESET output, active high (low version is available) can put MCU into reset if the voltage drops below threshold value and keep it there until the voltage is back or for at least 200ms. The PIC MCU has reset as input only and bi-directional feature is not used. The reset accuracy is ±2%.

**Watchdog timer.**
The watchdog circuit is used to monitor the MCU activity and trigger a reset. The MCU must trigger the WDI input with a high-low or low-high transaction every 1.6s. When the timer expire the reset input is triggered high and low every 1.8s. Timer is disabled in reset. The watchdog feature of the current circuit can’t be disabled.
**Power-fail comparator.**
The comparator is used to signal a failing power supply. An active-low PFO can be triggered by two conditions:

- By the PFI input voltage, compared to internal 1.237V reference voltage, $V_{PFT}$ ($PFI < V_{PFT}$).
- By $V_{CC}$ below $V_{SW}$ ($V_{CC} < V_{SW}$).

The comparator is a separate circuit and can be used to monitor even negative voltages. When not used, the PFI should be connected to ground and PFO unconnected.

**Component selection.**
Generic ceramic capacitors are placed at $V_{BATT}$ and $V_{OUT}$ with voltage above 10V. Used capacitors are 0.1uF Murata GCM319R71H104KA37D, in 1206 case for 0.041€/piece.

Resistors R1 and R2 are used to set the PFI threshold, the resulting voltage are compared to 1.237V. The current through resistors should be at least 1uA. Trip point is calculated using the formula:

The desired trip point is above the 0.9V but it will always trigger PFO, the DC-DC converter minimum input voltage. R2 is guessed to 1MOhm and 360kOhm, which gives a $V_{trip}$ of ca 1.9V.

The circuit is going to be replaced due to low current throughput. A 100mA is clearly not enough.
Data storage.

General description.
The next problem to solve is data storage. The system produces a lot of events that must be saved for a long time. Months. Years. And be easily accessed or moved to another system or read from computer. The following option was taken as possible.

- Serial EEPROM or SRAM
  EEPROM is the easiest way to add extra memory space to board and requires small amount of pins to work. A disadvantage is a relatively low size and storage optimization. Storage structure must be known in advance and migration is difficult. A read of complete EEPROM is not possible by computer and requires USB or Ethernet connection. Migration to another system is complicated and involves de-soldering (DIP packages are not taken in account and sockets for SOIC are expensive). EEPROM have sizes up to 1Mb and can be combined on single bus.

- USB stick
  A MCU with embedded host can use a USB memory stick to save log files. A framework is available at Microchip website for USB routines. The various sizes are possible. The solution was rejected because of board space use and USB implementation limitations. Otherwise, a migration is easy and solution is user-friendly as USB presents on every computer.

- Memory disk drive.
  A memory disk like SD Card or Compact Flash card can be accessed by MCU via SPI or parallel interface. A framework is available for basic functions. Cards are available in different sizes and volumes. An external card reader can be used on computers without. Another advantage is lower than USB power consumption.

  An SD Card was accepted with connector on board.

Secure Digital Card overview.

Secure Digital is one of many multimedia cards, but has some differences. The card is asymmetrically shaped and can’t be inserted wrong, electrical contacts are under the card surface and are protected, the card can be write-protected using a switch on card. Card detection mechanism is also presented.

Cards are usually formatted in FAT16, FAT32 or exFAT file systems.
The electrical interface is rather simple. Card requires supply voltage of 3.3V and 6 pins, the current requirements are 100mA while active, a good choice for power saving applications. The clock rate for a low-speed mode is 400 kHz, higher clocks are 25MHz and 50MHz.

The SD card can be accessed by four different interfaces, the parallel interface for faster operations or SPI for MCUs. SD 1-bit protocol, SD 4-bit protocol and SD 4-bit (is mandatory for high-speed cards) mode are not interesting as it is not supported by the framework.

Card is available in three sizes: the size of standard SD card is only 32x24 mm, the mini size of 21.5x20mm and micro size of 15x11mm. A standard size card is used.

Pin descriptions are shown in a table below:

<table>
<thead>
<tr>
<th>Pin #</th>
<th>Name</th>
<th>Type</th>
<th>SPI Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CS</td>
<td>I</td>
<td>Chip select (active low)</td>
</tr>
<tr>
<td>2</td>
<td>Data In</td>
<td>I</td>
<td>Host to Card Commands and Data</td>
</tr>
<tr>
<td>3</td>
<td>VSS1</td>
<td>S</td>
<td>Ground</td>
</tr>
<tr>
<td>4</td>
<td>VDD</td>
<td>S</td>
<td>Supply voltage</td>
</tr>
<tr>
<td>5</td>
<td>CLK</td>
<td>I</td>
<td>Clock</td>
</tr>
<tr>
<td>6</td>
<td>VSS2</td>
<td>S</td>
<td>Ground</td>
</tr>
<tr>
<td>7</td>
<td>Data Out</td>
<td>O</td>
<td>Card to Host Data and Status</td>
</tr>
<tr>
<td>8</td>
<td>RSV</td>
<td>I</td>
<td>Reserved</td>
</tr>
<tr>
<td>9</td>
<td>RSV</td>
<td>I</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Table 14: Pin Assignments in SPI Mode.

Where: S=power supply; I=input; O=output.

The electrical schematic requires external pull-up resistors on reserves lines to avoid unexpected current consumption of floating inputs. Two pull-ups are also placed on Card Detect (CD) and Write Protect (switches). Additional pull-up resistors are set on command and data lines (DataIn, DataOut, and CS) to protect inputs when no card is inserted or inputs

---

1 (SanDisk, 2003), Table 3-2. SPI Bus Mode Pad Definition.
are in high-impedance mode. Pull-up values for data and command lines are 10k - 100kOhm².

Card is powered and uses 3.3V interface.

**Microchip MDD File System Interface Library**

MDDFS was created to help users manipulate files on Secure Digital and Compact Flash files. SD cards are limited to SPI interface, for CF cards parallel interface is possible as well. Library is a part of a bigger library called Microchip Applications library that must be installed to compile the project.

There are two major help files:

- MDDFS Library Help.chm, placed in .\Microchip Solutions\Microchip\Help (Microchip, 2008)
- AN1045 Implementing File I/O Functions Using Microchip’s Memory Disk Drive File System Library in .\Microchip Solutions\Microchip\MDD File System\Documentation (Microchip, 2008).

A major part of information is taken from them and that is why references to specific parts are not shown.

During operations card shouldn’t be removed.

**Directory structure and includes.**

The library is divided into source and header files that are places in different directories. There are, however, additional dependencies of other library files and search parts should be added to IDE.

Here are some paths to add to the IDE search paths and to IDE file browser (include in project).

| \Microchip Solutions\Microchip\MDDFS Library Help.chm | Documentation and help files. Add to “Other” files section. |
| \Microchip Solutions\Microchip\MDD File System\Documentation\01045b.pdf | |
| \Microchip Solutions\Microchip\Include | Header files required by all libraries. |
| \MDD File System | Header files for library. |
| \PIC18 salloc | Header files for dynamic memory allocation |
| \Microchip Solutions\Microchip\MDD File System | Source files for library. |
| \Microchip Solutions\Microchip\PIC18 salloc | Source file for dynamic memory allocation. |

**Table 15: MDDFS include paths.**

Upon the successful configuration, the only include row needed to start library is:

² (SanDisk, 2003) page 3-8, table 3-5. Signal Line’s Load.
Other files, like CF part, are still included into the project, but are not used and compiler optimizations successfully remove unused code bits.

**Memory and hardware requirements.**

Memory usage of the library is quite large and requires additional storage for heap file and file system buffers.

Original linker script is copied into the project folder and modified with following rows (GPR 7 – 10 are replaced).

```c
DATABANK NAME=buffer1 START=0x700 END=0x8FF PROTECTED
DATABANK NAME=buffer2 START=0x900 END=0xAFF PROTECTED
```

Later, in CONFIG section, appropriate name is set by:

```c
SECTION NAME=dataBuffer RAM=buffer1
SECTION NAME=FATBuffer RAM=buffer2
```

In case of dynamic memory allocation, heap size may need to be changed and defined (GPR 8):

```c
SECTION NAME=_SRAM_ALLOC_HEAP RAM=gpr8
```

IDE should be configured to use a “Multi-Bank Model” (Project > Build Options > C18).

<table>
<thead>
<tr>
<th>Functions Included</th>
<th>Program Memory (C18)</th>
<th>Data Memory (C18)</th>
</tr>
</thead>
<tbody>
<tr>
<td>All extra functions disabled</td>
<td>11099</td>
<td>2121</td>
</tr>
<tr>
<td>(Read-Only mode)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>File search enabled</td>
<td>2098</td>
<td>0</td>
</tr>
<tr>
<td>Write enabled</td>
<td>7488</td>
<td>0</td>
</tr>
<tr>
<td>Format enabled (write must be enabled)</td>
<td>2314</td>
<td>0</td>
</tr>
<tr>
<td>Directories enabled (write must be enabled)</td>
<td>8380</td>
<td>90</td>
</tr>
<tr>
<td>Directories and search are both enabled</td>
<td>118</td>
<td>0</td>
</tr>
<tr>
<td>pgm functions enabled</td>
<td>288</td>
<td>0</td>
</tr>
<tr>
<td>FSpromptf enabled</td>
<td>2758</td>
<td>0</td>
</tr>
<tr>
<td>FAT32 support enabled</td>
<td>407</td>
<td>4</td>
</tr>
</tbody>
</table>

**Summa:**

|                        | 34950 | 2215 |

Table 16: MDDFS memory requirements.
The table above shows an approximate usage, compiler optimizations can significantly reduce size. Additional options like number of files opened and operations should be taken into account.

A MCU with at least 64kb memory is a suggested requirement for operations.

**Library configuration.**

Library configuration is done using two files. The easiest way to get them is a copy from SD demonstration folders, as they have little more than simple definitions. Configuration is done by commenting and un-commenting definitions in the files.

Note, that some MCU ports are multiplexed with ADC or programmable. Library doesn’t have any sense for it and it’s a user task to ensure that peripheral is set correctly (for example, SPI2 peripheral pin are programmed).

File `FSConfig.h` contains library configuration.

- **File system options.**
  - `FS_MAX_FILES_OPEN` – maxim number of files open.
  - `MEDIA_SECTOR_SIZE` – sector size, often unchanged.
  - `SUPPORT_FAT32` – support for FAT32 file system.

- **File management options to enable features.**
  - `ALLOW_FILESEARCH` – file search (functions FindFirst and FindNext).
  - `ALLOW_WRITES` – enable any writes to card.
  - `ALLOW_FORMATS` – format card, requires write permission.
  - `ALLOW_DIRS` – directory use and management, requires write permission.
  - `ALLOW_PGMFUNCTIONS` – PGM functions, PIC18 only.
  - `ALLOW_FSPRINTF` – write a formatted string to file with FSPRINTF().

- **Time clock configuration to generate timestamps.** Only one can be used at time.
  - `USEREALTIMECLOCK` – use an embedded real-time clock module.
  - `USERDEFINEDCLOCK` – use a static timestamp.
  - `INCREMENTTIMESTAMP` – use a static timestamp and increase it with 1 every time file accessed.

- `FS_DYNAMIC_MEM` – to use dynamic memory allocation for files. Requires heap and salloc functions (PIC18 only).

Current MCU has an internal Real-time clock and it is used to generate timestamps for files. All options, except format, are enabled, sector size and number of open files (default 2) are unchanged.
File `HardwareProfiles.h` configures the hardware on board. File consists of defines for all possible interfaces and cards, they are left unchanged or removed. File consists of a number of pre-processor directives for Microchip demo boards, they can be removed.

Following things must be defined correctly.

- `GetSystemClock()` set to system clock without any letters (8MHz is not accepted), to regulate correct clock. Compiler will fail if a system clock is too low (<400kHz).
- `USE_SD_INTERFACE_WITH_SPI` — set library to use SPI interface and SD card.
- Define Chip Select, Card Detect, Write Enable pins for card.
- Define I/O PORT / TRIS / LATCH registers for SPI.
- Define SPI registers for hardware SPI module.

**Library use.**

File management is very primitive and C-like. There are two important structures to be aware of.

1. `FSFILE` used to hold information about the file and is required by virtually all functions.
2. `SearchRec` — contains information about found files in search.

The initialization of the library is typically done by first checking a card presence in socket using `MDD_MediaDetect()` function and calling `FSInit()` function afterwards.

Read and write functions allow basic operations with files like reading n-bytes and should be enough for application.
Wireless module.

Selecting a wireless module.
A wireless module should have a set of features required to be accepted. Rather than creating an own module from existing IC (f. ex. Freescale MC1322X platform) the ready solutions was chosen. The main reason for that is the size and time required to start development with module.

For example, a Freescale is a 99-pin Platform-in-Package (PiP) 32-bit ARM7 core based MCU with integrated 2.4 GHz radio frequency transceiver and hardware acceleration in LGA (Land Grid Array) package. Unfortunately, with the landing pad of 9.5 x 9.5 mm makes it unavailable for home use. The major part of RF chips share the same disadvantage – they are very small.

Following requirements are applied:

1. Use a free radio band inside European Union and Sweden.
2. Have limited transmitter power. The power is usually limited by government.
3. Transmitter power should be enough to transmit data to at least 10 meters.
4. Module should be certified.
5. Have a relatively big size for surface mounting.
6. Easy and fast interface (SPI is preferred).
7. Easy to get.
8. Cheap.
9. Low power consumptions (sleep mode is preferred).

The transceivers can be generally divided into two categories based on used band:

- Sub GHz, operating near the 434/868/915 MHz bands.
- GHz, including wireless LANs at 2.4/5 GHz bands.

The GHz bands were chosen despite some of requirements because of it is assignment for data communication.

In the 2.4GHz band (5GHz is quite unusual right now) the choice is limited to Bluetooth, Wi-Fi or ZigBee modules.

The Bluetooth meets almost all requirements, but is ready modules were expensive.

The Wi-Fi modules have high power consumption, high data rates, that will be unused most of the time.

---

http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=MC13202
http://e-tjanster.pts.se/radio/frekvensplanen/Service.aspx
That leads to ZigBee. This was designed for low current consumption and data rates. The big advantage is its ability to build MESH and P2P networks.

The competitors were:

1. CIRRONET - ZMN2405HP-C – MODULE
2. XBee® and XBee-PRO® ZB SMT
3. Microchip MRF24J40MA

The Microchip MRF24J40MA was chosen because of its low price, size and framework.

**Wireless transceiver Microchip MRF24J40MA**

The MRF24J40MA is a MRF24J40-based 2.4 GHz IEEE Std. 802.15.4 compliant, surface mount module with PCB antenna and required circuits like internal voltage regulator and integrated crystal. The module itself can be used as it is, but it becomes more useful with supplied framework.

The module can be used with ZigBee, MiWi or MiWi P2P protocols. Migration between protocols is rather easy with supplied interfaces and will be covered later.

**MRF24J40**

The original MRF24J40 MCU has a lower price than ready module. The package type is QFN (Microchip).

<table>
<thead>
<tr>
<th>Amount</th>
<th>Price (USD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1–25</td>
<td>3.24</td>
</tr>
<tr>
<td>26–99</td>
<td>2.97</td>
</tr>
<tr>
<td>100+</td>
<td>2.69</td>
</tr>
</tbody>
</table>

**MRF24J40MA**

A slightly larger solution is a MA (or MB for higher power), soldered on PCB and includes antenna, matching circuit, crystal and power management.

<table>
<thead>
<tr>
<th>Amount</th>
<th>Price (USD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1+</td>
<td>9.95</td>
</tr>
</tbody>
</table>

---

5 http://www.rfm.com/products/zigbee.php
6 http://www.digi.com/products/wireless/zigbee-mesh/
Except the usual SPI pins the chip has WAKE, RESET and INTerrupt pins.

**Wireless framework (MiWi P2P).**

Since a module is a transceiver only, it can be programmed to be used with different protocols within 802.15.4 specification.

Microchip offers three protocol options (or use IC directly)

- ZigBee, memory use app. 100K, unlimited hops.

MiWi, a lighter version of ZigBee, divided into parts, only one can used at time.

- MiWi Mesh, Star/Mesh topology, app. 20K memory use.
- MiWi P2P, Star/P2P topology, app. 10K memory (3K with lower functionality).

Although, the memory usage depends on protocol configuration, optimizations, security and device type. Memory optimization can reduce memory usage to 50K (Microchip). Still, the use of ZigBee protocol is superfluous for its size. MiWi P2P was chosen to be used mostly because of simplicity.

Features of special interest:

- Sleeping device at the end of communication and indirect messaging.
- Energy detect scan to choose channel.
- Active scan to join existing networks.
- Channel hopping.

Protocol is described in applications notes from Microchip and help file of the application library. A short overview of features and configuration is presented below.

**Device types.**

Devices in a network are divided by functionality and function.

A Full Function Device (FFD) is typically always on and powered from the main source. A Reduced Function Device (RFD) is powered from the battery and can be turned off.

The main device in the network is a Personal Area Network (PAN) Coordinator, is a FFD device that starts the network and waits for other devices to join. An End Device is either FFD or RFD and starts after the PAN device.

**Network structure.**

MiWi protocol can form three network topologies.

- Star topology.
The network coordinator initializes communication and accepts connection. Network can have multiple end devices, but they can only communicate to coordinator or through it (end to end is possible). There can only be one network coordinator.

- Peer-to-peer (P2P) topology.

The P2P topology allows additional coordinators to be added to the network. The PAN coordinator is still only one. Nodes can only connect to one coordinator at time. Since the network is tree-like, messages can be sent by many ways to the end device. The number of hops defines how “wide” a tree is.

- MESH topology.

MESH topology is similar to P2P, but tree structure doesn’t have to be followed.

As topology suggests, the P2P protocol doesn’t have a routing mechanism and communication is limited to radio range.

MiWi realisation supports only non-beacon networks. Device can transmit data at any time. Energy consumption of FFD is increased because radio is turned on all the time, but RFD’s consumption is lower due to lower traffic.

Since the RFD radio can be turned off during inactivity, there should be a way to deliver a message to an idle device. That feature is called indirect messaging. In that case, the FFD must save it until the end device radio is turned on by a key press or timer.

An end device can also make a broadcast to all other nodes in the network.

**Network Addressing**

MiWi network uses three types of addresses.
- A long - Extended Organizationally Unique Identifier (EUI-64), unique all around the world.
- A short, unique address assigned by the PAN coordinator or parent in the network (2 bytes).

Since the P2P network is one-hop network, the short address is only used by the stack and for broadcast messages.

MiWi MESH protocol has an additional address to define a group of nodes, PAN Identifier (PANID) - the address is same for all nodes and is assigned by the user.

The EUI address is shared with the Ethernet module (see the Ethernet section) and is taken from the Microchip MAC EEPROM. The EEPROM is also used to store the network data with Network Freezer in case of power down. More information is available in AN1204 - Microchip MiWi™ P2P Wireless Protocol (Microchip, 2010) and AN1066 - Microchip MiWi™ Wireless Networking Protocol Stack (Microchip, 2010).

**Program interfaces.**

Microchips wireless application stack includes a number of interfaces. The higher level consists of MiApp interface, the application interface common for all Microchip protocols. MiMac interface is used to bind the application level interface with hardware receivers, supported by MiMac. In other words, MiApp is used to choose the protocol and MiMac to choose the transceiver. Both layers make protocols and transceivers interchangeable.

![Figure 32: Microchip wireless MiWi stack diagram (Microchip, 2009).](image-url)
MiApp is described in details in AN1284 - Microchip Wireless (MiWi™) Application Programming Interface – MiApp (Microchip, 2009). The interface includes protocol configuration in a configuration file and signatures of function calls. There are five categories of functions in MiApp:

- Initialization using the configuration files.
- Hand-shaking, network discovery.
- Interface to send messages.
- Interface to receive messages.
- Special functionality.

The configuration file includes both hardware configuration and software definitions.

MiMac is described in AN1283 - Microchip Wireless (MiWi™) Media Access Controller – MiMac (Microchip, 2009). Layer provides a basic addressing and data functionality above the raw hardware. Layer can be used without MiApp.

**Directory structure and includes.**

Here are some paths to add to the IDE search paths and to IDE file browser (include in project). The Transceivers directory includes definitions of hardware transceivers and Wireless Protocols.

<table>
<thead>
<tr>
<th>Directory Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\Microchip Solutions\Microchip\MiWi DE Help.chm (Microchip, 2009)</td>
<td>Documentation and help files. Add to “Other” files section.</td>
</tr>
<tr>
<td>\Microchip Solutions\Microchip\Include \Transceivers</td>
<td>Header files required by all libraries.</td>
</tr>
<tr>
<td>\Microchip Solutions\Microchip\Include \WirelessProtocols</td>
<td>Header files for transceivers.</td>
</tr>
<tr>
<td>\Microchip Solutions\Microchip\Transceivers</td>
<td>For protocols.</td>
</tr>
<tr>
<td>\Microchip Solutions\Microchip\WirelessProtocols</td>
<td>Source files for transceivers.</td>
</tr>
<tr>
<td>\Microchip Solutions\Microchip\WirelessProtocols</td>
<td>Source file for protocols.</td>
</tr>
</tbody>
</table>

*Table 17: MiWi include paths.*

The configuration files should be created and included into the project:

```c
#include "ConfigApp.h "; // Protocol configuration
#include " HardwareProfile.c "; // Hardware initialization (pin, register)
#include " HardwareProfile.h "; // Protocol hardware.
```

**Memory and hardware requirements.**

MRF24J0MA Module requires three additional pins (RESET, INT, and WAKE) except the standard four SPI. The memory requirements are as low as 3Kb. Hardware SPI is not necessary due to embedded software emulation. That makes it possible to use module with virtually any MCU with 14+ pins.
<table>
<thead>
<tr>
<th>Feature</th>
<th>Count</th>
<th>Memory Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable Intra-PAN Communication</td>
<td>462</td>
<td>0</td>
</tr>
<tr>
<td>Enable Sleep</td>
<td>186</td>
<td>0</td>
</tr>
<tr>
<td>Enable Security (Without Frame Freshness Checking)</td>
<td>500</td>
<td>48</td>
</tr>
<tr>
<td>Enable Security (With Frame Freshness Checking)</td>
<td>1488</td>
<td>54</td>
</tr>
<tr>
<td>Enable Active Scan</td>
<td>1070</td>
<td>69</td>
</tr>
<tr>
<td>Enable Energy Scan</td>
<td>752</td>
<td></td>
</tr>
<tr>
<td>Enable Indirect Message</td>
<td>950</td>
<td>Size * TX Buffer Size</td>
</tr>
<tr>
<td>Enable Indirect Message with Capability of Broadcasting</td>
<td>1228</td>
<td>Size * TX Buffer Size</td>
</tr>
</tbody>
</table>

**Summa:** 6636 171

Table 18MiWi P2P memory usage.

As seen, the most features can be disabled without any complications. Target size can be minimized to 3Kb with the TARGET_SMALL macro. In that case the communication between PANs and freshness security check are disabled.
Ethernet module.

Communication from the main board is made through Ethernet. And there are a few reasons why.

Firstly, the Ethernet is the most widely used network available everywhere. Ethernet is based on the standard that ensures reliability of network and data transmission. And finally, the range of communication is not limited.

The bandwidth speed is 10T, 100T or 1000T Mbs. 10Base-T is more interested for embedded devices and more than sufficient for most applications. Those devices are usually placed behind a switch or router. 100Base-T has a better performance and low latency is more suited for real-time operations.

Ethernet on embedded devices.

Adding Ethernet to embedded devices is complicated. Most of the Ethernet controllers are for PC use and have a large interface pin number, typically an 80 pin MCU is required. As a result, the PCB footprint is quite big. Another problem is insufficient program memory on a low segment MCUs.

A “real” transceiver like SMSC LAN83C185-JT in TQFP64 case is a good example of such controller. Most of controller pins are control and data wires. IC footprint is small, but soldering is complicated for home use. All these considerations make it useless for small embedded operations. The price is from 5.33€ (62.48SEK) to 2.66€ (31.54SEK), 500+.

Another option that has become available during the past years is SPI based controllers. Those have a lower pin count and many fewer interface lines between circuits. The SPI bus allows integration into early produced solutions. The footprint is reduces as well. The main consideration for that type of solution is program memory size (about 25K, depending on options). That solution has become number one and is currently used in design. More detailed description is in next section.

The next solution and the one that has been chosen and partly used, but was dropped later on is an MCU with embedded Ethernet controller like PIC18F97J60. The MCU is
in TQFP 64/80/100 case and the price from 5.67€ (64.80SEK) to 3.61€ (37.21SEK), 100+. That was considered as number one choice, but dropped due to of difficulties in PCB soldering.

**SPI base controller ENC28J60 (Microchip).**

ENC28J20 is a stand-alone SPI Ethernet controller with dual-port RAM buffer for transmitted and received packets. Controller has a MAC (Medium Access Control) logic that implements the 802.3 compliant MAC and a PHY (Physical Layer) logic that decodes signals on twisted pairs.

The controller is not alone in its series. Other controllers are ENC424J600 and ENC624J600 with slightly more advanced features and parallel interface.

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>ENC28J60</th>
<th>ENC424J600</th>
<th>ENC624J600</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAC</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>PHY</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>TX/RX RAM Buffer(bytes)</td>
<td>8192</td>
<td>24576</td>
<td>24576</td>
</tr>
<tr>
<td>Interrupt Pin</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>LEDs</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Op. Voltage (V)</td>
<td>3.3</td>
<td>3.3</td>
<td>3.3</td>
</tr>
<tr>
<td>Max. Speed (MHz)</td>
<td>25</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>Interface</td>
<td>SPI</td>
<td>SPI and Parallel</td>
<td>SPI and flexible parallel</td>
</tr>
<tr>
<td>Pre-Programmed MAC Address</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Security Engines</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Standalone Ethernet Controller</td>
<td>10Base-T</td>
<td>100Base-T</td>
<td>100Base-T</td>
</tr>
<tr>
<td>Packages</td>
<td>28 pin SOIC, SPDIP, SSOP, QFN (6x6 mm)</td>
<td>44-Pin (TQFP and QFN)</td>
<td>64-Pin (TQFP)</td>
</tr>
</tbody>
</table>

Table 19: Stand-alone Ethernet controllers (Microchip).

A different variation of packages makes ENC28J60 favourite for hobby use. A lack of pre-programmed MAC address requires an additional IC like Microchip MAC EEPROM chip.

Prices are shown below:

<table>
<thead>
<tr>
<th>Amount</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 – 9</td>
<td>3.00€ (38.49 SEK)</td>
</tr>
<tr>
<td>10 – 99</td>
<td>2.67€ (25.72 SEK)</td>
</tr>
<tr>
<td>100 +</td>
<td>2.15€ (22.56 SEK)</td>
</tr>
</tbody>
</table>

Table 20: ENC28J60 price.

Price of other controllers is slightly higher.
A SPI communication except the standard SPI pins requires additional interrupt pin (INT) and RESET (not shown). Reset can also be performed by an SPI command. Another useful feature is a programmed CLKOUT pin that can be used to clock an external device.

A quite unusual feature is that the controller has 5 power pairs (10 of 28 pins are used to power IC). A full set of features and detailed information can be obtained in ENC28J60 Datasheet (Microchip, 2008).

External components and PCB considerations.
The controller requires additional external components to interface the twisted signal line. A PHY layer needs a 2.32kOhm, 1% resistor to be placed as close as possible to the RBIAS pin with low signal interference.

Core logic is operating from the nominal 2.5V and uses on-chip generator to step-down the input voltage. A 10uF low-ESR capacitor (tantalum or ceramic type) is placed at $V_{cap}$ pin. Both interfaces have two 49.9Ohm 1% resistors and a 0.1uF capacitor to properly terminate the transmission line and minimize signal reflections. A ferrite bead rated at least 80mA is places at the TPOut interface. Ethernet transformers and 75Ohm resistors are currently not placed on the board, but a RJ-45 connector with integrated magnetic is used instead. LEDs are usually integrated into connector as well.

Because of the high currents flowing through the wires, all power pins should be connected direct to the same power source and star-ground configuration should be used. Each power pair requires its own decoupling capacitor, placed close to the pins.
LED polarity is automatically detected on reset and can be configured by software as well.

A STEWART CONNECTOR SL-60062-F RJ45 jack with integrated magnetic costs from 5.19€ (51.87 SEK)/piece to 3.91€ (38.37 SEK)/100+ pieces.

VISHAY DRALORIC CRCW120649R9FKTA 49.9Ohm 1% resistors are used.

**Microchip TCP/IP stack.**

The ENCJ60 controller can be used for raw Ethernet communication. But it can gain more power with Microchip TCP/IP stack from Microchip software libraries (although there are other implementations for Microchip MCUs). Stack supports a full set of microcontrollers, including PIC18F, PIC24, PIC32 and dsPIC families. Unlike the other stack Microchip has provided a number of software to generate the configuration and images located in \Microchip Solutions\Microchip\TCP/IP Stack\Utilities Directory.

<table>
<thead>
<tr>
<th>TCP/IP Configuration Wizard</th>
<th>TCP/IP configuration wizard generates stack configuration file.</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPFS2 Utility</td>
<td>MPFS2 utility to create image from web pages and remove unnecessary information from files (minimizes size).</td>
</tr>
<tr>
<td>Microchip Web Preview Tool</td>
<td>Web preview tool for MPFS2 utility, shows images as they will appear in browser.</td>
</tr>
<tr>
<td>Hash Table Filter Entry Calculator</td>
<td>Hash table calculator, not used in this project.</td>
</tr>
<tr>
<td>Microchip TCP/IP Discoverer</td>
<td>A program that discovers embedded devices using the broadcast packet and Announce protocol.</td>
</tr>
<tr>
<td>MCHPDetect</td>
<td>Another Announce protocol example, not used in the project.</td>
</tr>
</tbody>
</table>

*Table 21: TCP/IP stack software.*

The help files are:

- Microchip TCP/IP Stack help.chm, placed in .\Microchip Solutions\Microchip\Help (Microchip, 2010).
- AN833 - The Microchip TCP/IP Stack, (Microchip, 2008).
- Web seminar:
  TCP/IP Networking: Web-Based Control (Microchip, 2007).

**Stack architecture.**

The stack has a modular architecture with high degree of abstraction. Each layer accesses layers below it that can run independently on actions like timeout or packet arrival.
Figure 38: TCP/IP stack architecture.

Each layer is implemented in separate source files. There are some differences in Microchip realisation as compared to TCP/IP reference model. Stack layers can access layers that are not directly below it. Stacks needs to perform actions asynchronously (like timeouts) and stay independent from the main program. Unlike the higher level MCUs the 8-bit microcontrollers are not really suited for multi-tasking. A technique called cooperative multitasking is therefore used. Each task performs its job and returns the control to the next task. Those tasks are called StackTask and ARPTask. Microchips realization uses its own cooperative multitasking and is not dependent from the system realization. However, the host system must use a cooperative multitasking and divide other tasks into the functions or use FSM. That approach makes stack less dependent from the main application and new modules can be used without big code changes.

**Directory structure and includes.**

Stack has a more complicated structure, compared to other libraries and includes depends on used layers.

<table>
<thead>
<tr>
<th>Directory Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\Microchip Solutions\Microchip\TCP/IP Stack Help.chm</td>
<td>Documentation and help files. Add to “Other” files section.</td>
</tr>
<tr>
<td>\Microchip Solutions\Microchip\Include</td>
<td>Header files required by all libraries.</td>
</tr>
<tr>
<td>- \TCP/IP Stack</td>
<td>Header files for library.</td>
</tr>
<tr>
<td>\Microchip Solutions\Microchip\TCPIP Stack</td>
<td>Source files for library.</td>
</tr>
</tbody>
</table>

Table 22: TCP/IP stack include paths.

There are additional files every application should include into the project (configuration files must be created.

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARP.c, ARP.h</td>
<td>To discover MAC address associated with IP.</td>
</tr>
<tr>
<td>Delay.c, Delay.h</td>
<td>Delays for stack functions (blocking).</td>
</tr>
<tr>
<td>Helpers.c, Helpers.h</td>
<td>Helpers to stack functions.</td>
</tr>
<tr>
<td>IP.c, IP.h</td>
<td>Internet layer functionality.</td>
</tr>
<tr>
<td>StackTsk.c, StackTsk.h</td>
<td>Callbacks for cooperative multitasking.</td>
</tr>
<tr>
<td>Tick.c, Tick.h</td>
<td>Stack timers.</td>
</tr>
<tr>
<td>MAC.h</td>
<td>Implementation of MAC layer.</td>
</tr>
<tr>
<td>TCP/IP.h</td>
<td>Main include file.</td>
</tr>
</tbody>
</table>

Figure 39: TCPIP stack includes.
Additional includes are hardware configuration of transceiver, hardware configuration of the board `HardwareProfile.h` and stack configuration file `TCPIPConfig.h`.

The project main file should have a declaration:

```c
#define THIS_IS_STACK_APPLICATION
```

The source files included in the main file are:

```c
#include "StackTsk.h"
#include "Tick.h"
```

If other modules, e.g. http, requires additional includes (typically one row per module).

**Stack configuration.**

Stack configuration is placed in different files.

Hardware configuration includes:

1. General pin and register setting for the system and clock macro.
2. External storage configuration: definitions for pins and registers for selected web storage (selected in `TCPIPConfig`).
3. Transceiver configuration: SPI pins and registers for selected module.

Address configuration includes MAC and IP addresses. MAC address is obtained from the MAC Address chip or can be hardware defined in code. IP address can be obtained by DHCP, Auto-IP or static address set at compile time.

Stack configuration is generated by the `TCPIPConfig` application from existing sources in application library. Utility defines most commonly used variables, placement for web pages and modules.

**Stack initialization.**

Stack must be initialized in code. That is done by calling:

```c
TickInit();
StackInit();
```

Additional modules are initialized after by calling corresponding functions.

Once stack is initialized, the cooperative multitasking is started inside the loop.

```c
// Update tick count. Can be done via interrupt.
TickUpdate();
// Let Stack Manager perform its task.
StackTask();
// Let any Stack application perform its task.
HTTPServer(); // Only if HTTP is used.
```
The TickUpdate and StackTask should always present.

**Memory usage.**

Stack memory usage is dependent on used modules. These values below are for reference only. The amount of used memory is dependent of application and realization.

<table>
<thead>
<tr>
<th>Module</th>
<th>Program memory (words)</th>
<th>Data memory (words)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAC (Ethernet)</td>
<td>906</td>
<td>5</td>
</tr>
<tr>
<td>SLIP</td>
<td>780</td>
<td>12</td>
</tr>
<tr>
<td>ARP</td>
<td>392</td>
<td>0</td>
</tr>
<tr>
<td>ARPTask</td>
<td>181</td>
<td>11</td>
</tr>
<tr>
<td>IP</td>
<td>396</td>
<td>2</td>
</tr>
<tr>
<td>ICMP</td>
<td>318</td>
<td>0</td>
</tr>
<tr>
<td>TCP</td>
<td>3323</td>
<td>42</td>
</tr>
<tr>
<td>HTTP</td>
<td>1441</td>
<td>10</td>
</tr>
<tr>
<td>FTP Server</td>
<td>1063</td>
<td>35</td>
</tr>
<tr>
<td>DHCP Client</td>
<td>1228</td>
<td>26</td>
</tr>
<tr>
<td>IP Gleaning</td>
<td>20</td>
<td>1</td>
</tr>
<tr>
<td>MPFS</td>
<td>304</td>
<td>0</td>
</tr>
<tr>
<td>Stack Manager</td>
<td>334</td>
<td>12 + buffer</td>
</tr>
<tr>
<td><strong>Total:</strong></td>
<td><strong>10,686</strong></td>
<td><strong>156</strong></td>
</tr>
</tbody>
</table>

Table 23: TCP/IP stack memory usage.

Still, the huge memory is taken by the MPFS module that stores a compiled image of web pages served by HTTP2 Server module. For a more or less advanced web page those numbers are ridiculous.

A 32kB memory should be enough for simple tasks. The 8-bit MCU are available up to 128kB, so there are some margins.

**Stack API modules.**

The stack consists of a set of modules. Some of them provide a general TCP/IP functionality and are configured on compile time.

Those modules are used in the project, but have a low grade of user interaction or are transparent to the user.

- **Announce** – device discovery in DHCP network by UDP protocol.
- **ARP** – address resolution protocol, a layer of TCP/IP.
- **DNS Client** - Domain Name Service associates host name with IP address.
- **Dynamic DNS Client** – binding of dynamic IP address to a public DDNS service.
- **ICMP** - Internet Control Message Protocol is useful to check the device state by ping command.
**NBNS** - NetBIOS Name Service protocol is like DNS, but for local networks. Device can be accessed by its name (like BOARD) instead for IP address.

**Performance tests** – embedded benchmark to test TCP and UDP performance.

**Reboot** – module allows rebooting a device by sending a command. Often used together with boot loader.

**TFTP** - Trivial File Transfer Protocol can be used to upload a firmware to a device and modify web pages on embedded web server.

**HTTP2 Module.**
The module implements a simple embedded web server. Server uses image provided by MPFS2 utility. Server has following features: dynamic variables, form processing, authentication, cookies and compression. Those sometimes work like its mature equivalents, but have some limitation.

There are two main files used by the server. The HTTPPrint.h is automatically generated by the MPFS2 utility for dynamic variables. The CustomHTTPApp.c is a user-generated file with current application behaviour.

Dynamic variables are inserted surrounded by a tilde, ~myVariable~. The MPFS2 Utility will automatically generate corresponding callback: void HTTPPrint_myVariable(void).
Unlike the usual functions, callback doesn’t return any value, but uses TCPPutROMString (or RAM) function to send the result. A more advanced output for array-like variables is done as: ~myVariable(1)~, ~myVariable(2)~, ~myVariable(3)~. An index is the taken by the call-back function to select the corresponding place. That way is limited to 16 bytes. For longer output a curHTTP.callbackPos should be used to track the output length.

A simple include feature is possible using ~inc:header.inc~. The statement will load the header.inc file at command location. Includes are not recursive, so dynamic variables in header.inc are not parsed. Usually such files are used to include the common elements to the page like header, footer or menu.

It is often required to get data from the user by the web form. The most commonly used methods are POST and GET requests. The GET data is visible in the URI; POST is more convenient for larger amount of information. Both methods are supported. Appropriate methods should be implemented in CustomHTTPApp.c.

Sometimes, it is necessary to restrict user access to pages. An Authentication with HTTPNeedsAuth function is possible. Module is not hardware coded and different combinations are possible. Callbacks are also implemented in CustomHTTPApp.c.
Cookies are another feature that allows storing data between the sessions. Small pieces of data are stored in the browser. The server needs to provide the way to invalidate the data after some period. Cookies are not used in this project.

Static files can be compressed for faster transmission. The compression is done by the MPFS2 utility by the file extension and can be configured in the application options. The compression is managed by the server and browser and is invisible for the end user. When the compression is used, the selected files are compressed using the gzip algorithm before and returned as an archive to the browser. The browser then extracts the files and shows content to the user. That is especially effective on the html files with lot of redundant tags and plain text files.

**MPFS2 Module.**

Except the generation project files and pages MPFS2 Utility is also in charge of storing the pages on MCU. An older FAT system is used for that. As mentioned before, pages are very large, as compared to the program memory of 8-bit MCU, so any garbage like comments, spaces is cleaned during the optimization.

Web pages can be placed in different places on the server. The easiest option is to store pages in embedded flash memory. Web pages are the compiled into a .C-file and stored as usual include file in the project. Any modification at runtime is not possible.

Pages can also be stored on external in external EEPROM (up to 1Mbit) or SPI Flash (up to 32Mbit). In that case, the web images can be uploaded using the HTTP or FTP protocols without recompiling the device. Space can be reserved for user application.

Placement is selected in the TCPIPConfig utility, web pages are converted to image with the MPFS2 utility.

**TCP Module.**

TCP modules allow simple communication to remote server. The module is used to send a SMS to a phone when needed. It is also possible to fetch a result from the server (return codes, text, and search results). A communication starts with **TCPOpen** and ends **TCPDisconnect** instructions. Data is send using the **TCPPutROMString** function (or its RAM alternative). Data is send using the Flush method.

```cpp
TCPPutROMString(MySocket, (ROM BYTE*)"GET ");
TCPPutROMString(MySocket, RemoteURL);
TCPPutROMString(MySocket, (ROM BYTE*)" HTTP/1.0\nHost: ");
TCPPutString(MySocket, ServerName);
TCPPutROMString(MySocket, (ROM BYTE*)"\nConnection: close\n\n");
```

The code above is used to connect to a remote server and request the URL using the HTTP GET. That is equivalent to submitting a form. The result is then fetched with the **TCPGetArray** function.
MAC address chips.
Both Ethernet and Wireless module requires a unique MAC address defined by IEEE that identifies the hardware and represents the actual physical address. There are two different addresses: a 48-bit EUI-48 and 64-bit EUI-64.

EUI Addresses are issued by IEEE and consist of:

- **OUI** – organisation unique identifier, a 24/36 bit value that is unique for every organisation assigned by IEEE.
- **EI** – 40/28 bit Extension Identifier, assigned by the organisation itself to every device to every device.

The standard way to obtain those codes is the IEEE. Unfortunately, the procedure is complicated and volume limited. A 16.7M codes costs 1.600$, a lowest amount is 4096 codes for 550$. Every code is then assigned and programmed to every unit. That requires a new program file for every MCU, increasing the programming costs. A more convenient way is to use the same image (in that case programming is done by the chip manufacturer at the factory, option is available at MicrochipDirect).

Another option is to use a company that will provide a smaller amount of chips in hardware. Microchip has a set of MAC EEPROM chips, an EEPROM memory with write-protected, unique programmed EUI-48 address. Communication is done using the SPI, I2C or UNI/O bus. The OUI is the same: 00-04-A3 and belongs to Microchip.

Different applications uses different addressed. EUI-48 is typically used by Ethernet, Bluetooth, 802.11 Wi-Fi, ATM and SCSI devices. EUI-64 is used by ZigBee, 802.15.4, MiWi, FireWire and IPv6 devices.

A EUI-48 address can be converted to EUI-64 by inserting a 16 bit FF-FF or FF-FE (hex) between the OUI and EI.

Example: 00-04-A3-12-34-56 becomes 00-04-A3-FF-FE-12-34-56.

![Figure 40: Suggested SPI configuration for MAC chip.](image)

The actual design uses the same MAC address for both Ethernet and MiWi communication (MiWi is converted to 64-bit). More information can be obtained at Microchip web seminar:
MAC Address Chips (Microchip) and http://www.microchip.com/MAC (Microchip). In the reality, two chips should be used to avoid interferences (information is unclear) and since both modules can use an EEPROM to store configuration.

At the moment, there are five chips available. Chips differ at operational frequency (100kHz, 400kHz and 10MHz, depend on voltage) and memory organisation. Operation voltage is 1.7 (1.8) to 5.5V. All chips are 2kBit.

![Figure 41: Microchip MAC chip packages.](image)

Chips are in SOT-23 and SOIC packages. Except the usual SPI pins, the WP and HOLD pins are present. The active low HOLD is used to pause the communication to the device, i.e. all communication on data wires is ignored. Active low WP – Write Protect is used to disable all writes to the memory. Ether pins are used and both are connected to the positive supply with resistor pull-ups.

The maximum clock frequency of 10MHz is only possible when the chip is powered from 4.5-5.5V. On the 3.3V system, the frequency is around 5MHz.

A more detailed description is available at 25AA02E48 Datasheet (Microchip, 2010). Additional information can be found at AN1040 - Recommended Usage of Microchip SPI Serial EEPROM Devices (Microchip, 2006).

Mac chips are rather cheap and easy alternative, as compared to the full block price.

<table>
<thead>
<tr>
<th>Amount</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 – 9</td>
<td>0.57€ (6.20 SEK)</td>
</tr>
<tr>
<td>10 – 99</td>
<td>0.49€ (3.75 SEK)</td>
</tr>
<tr>
<td>100 +</td>
<td>0.33€ (3.47 SEK)</td>
</tr>
</tbody>
</table>

Table 24: 25AA02E48 Price.
Making a web page.

A common technique to decorate the HTML files is Cascading Style Sheets. The main purpose of CSS is to separate the content of the document (HTML codes) from its representation: fonts, colours and layout. As a new technician it can be tricky to get a page one has in mind to look good (a correct word is impossible). The presence of many browsers and versions makes it extremely difficult to produce a result that will: one – look good; two – look good in different browsers. A usual approach in such a case is to hire a web designer to manage the styles and layout. Another option is to let someone else design the look and styles for a pre-defined HTML structure.

CSS frameworks.
A CSS framework is a set of external CSS styles that can be included into an existing project. Frameworks are usually based on some suggestions about HTML structure and rules. Because of content and view separation, the layout and appearance of a web page can be changed without changing the actual content. Another important task is to enable rapid development of the page by focusing on the structure and content without going into CSS details.

For this project of this paper, the following requirements are desired:

- Small size.
- Embedded Ajax or JavaScript engine (framework). The Ajax is especially good for embedded, slow, applications as it allows reloading only a part of the page and making some interactive actions like sorting.
- Easy to learn, good documentation and examples.
- Different page layouts: one, two three columns.
- Browser support, active development.
- Licensing for private and commercial use.

A quick search gave a list of promising competitors:

Grid frameworks: a grid framework divides a page into an imaginary net of equal size (about 24). A designer then uses that to make an offset, set width and placement. Those frameworks are usually using the fixed page width. A good overview of a grid system can be found in Web designer depot article: Fight Div-it is and Class-it is With the 960 Grid System (Fight Div-itis and Class-itis With the 960 Grid System., 2010).

Example of Grid frameworks:

- Blueprint is a CSS framework (http://www.blueprintcss.org/), is a very easy to learn and basic framework.
- 960 Grid System (http://960.gs/), another framework with impressive results.

YAML was chosen for this project as a very beautiful, promising and flexible alternative. The documentation is excellent and the folder structure makes updates easy.

### YAML - Yet Another Multicolumn Layout CSS framework.

YAML is a framework for a flexible column and grid based layouts. The framework is distributed under the Creative Commons Attribution 2.0 License, which allows even commercial use under conditions of link back to the framework page and authors name.

The framework uses a Top-Down principle of design, where the programmer start with a skeleton-layout with most used basic elements.

![Figure 42: YAML skeleton (from manual).](image)

The skeleton of YAML consists of a page, header and footer and three columns. The column order can be changed, but the general rule is that the middle, third, column has a floating width. Columns one and two have a fixed width. Still, YAML is designed for flexible layouts and virtually any element can be removed and customized.

YAML has separated user and framework CSS files. The framework files are placed in `yaml` directory and should not be modified. A user-defined CSS are in the `css` folder and are
automatically linked (there are some rules). Because of that structure, the framework updates do not require any code change.

**Folder structure.**
The framework has additional folders. As mentioned earlier, the main YAML files are placed in the `yaml` folder and the user CSS in the `css` folder. A documentation folder includes a full documentation (online documentations is a perfect example of all features) in English and German. The `examples` folder includes a layout and style examples. JavaScript framework jQuery and other JS files are placed in `js` folder. Additional tools for creating layouts are in `tools` folder.

**Components.**
There are a number of optional components; included by a simple CSS include statement.

- Navigation: sliding doors, shiny buttons, and vertical list navigation.
- Adjustments for printing media. The decorations like navigation, footer and other elements not usable on paper are not printed.
- Form construction kit. Creates a more advanced and functional forms compared to usual HTML. Labels and style are added and an error reporting (must be implemented by programmer) is possible. Form layout can be changed using the JS.

A full framework review is not possible here. Complete information is available on YAML website: [http://www.yaml.de/en/documentation/](http://www.yaml.de/en/documentation/) (Jesse, 2010) or inside the documentation folder.

The framework is actually rather large in size for an embedded application, but all its features make it hard to resist.
Sending an SMS: SMS Gateway provider.

The main board needs to send a SMS message from time to time. The most common approach in that case is to use an external GSM transceiver to connect to the mobile network. Another approach is to use the internet and a web-service that allows sending a message. That kind of services is called SMS Gateways.

For this project the (Clickatel) gateway was selected for good application interface and existing integration to surrounding devices. The other gateways have similar possibilities.

The possible ways to send a message are:

- **HTTP/S** is the easiest and the one used here. The submission is very like the standard HTTP form submission using a GET method. The only difference is that + (plus) is used to replace spaces instead of %20 character. The submitted message looks like:
  

  The username, password and api_id are bound to Clickatel account and private. The receiving telephone number is in the TO field and a message text is in a TEXT.

- **SOAP** is a XML-based protocol, less suited for embedded applications.

- **FTP** is another easy solution. The message is uploaded to an ftp account as a text file.

- **SMPP** is a standard for peer-to-peer messaging.

- **SMTP** (Email to SMS) is send using an Email to sms@messaging.clickatell.com with account details above.

- **COM Object** is a solution for Windows applications and ASP pages.

- **XML** is a more structures way to send a message, a little more complicated on embedded systems, but still possible.

The methods above use a similar data structure and API can be changed easily. The HTTP, FTP, SMTP, XML API can be used by the system.
Summary and conclusions.

The project was thought as a 10 week thesis. The schedule was very tight and pressed from the beginning. From the current point of view, the project should take about 12 weeks of pure construction time plus one week for software design. This is said based on the achieved knowledge of faults and how components work.

The biggest surprise was the PCB part. The original idea was good, but unfortunately failed due to manufacturing difficulties in the limited home environment. The potential replacements generally have a larger footprint and price. During the project time, every module has been tested individually in order to find out if it actually worked according to expectations. The assembly unfortunately couldn’t be produced without PCB errors. Although the software libraries have been individually tested as such, the integrated software of the complete system has not been tested due to the hardware problems.

The total price of the system was calculated using the DesignLink interface in Eagle CAD. Because of the presence of test pin headers the price is a higher. The component price for the main board was calculated to 70€ for a single board or to 40€ for 100 boards.

In the future, the project will be revised and probably, moved to a 16-bit architecture.
References.

Bibliography


Microchip Technology Inc.


Microchip. (2009). AN1283 - Microchip Wireless (MiWi™) Media Access Controller – MiMAC.


Microchip. (n.d.). MRF24J40 Data Sheet IEEE 802.15.4™ 2.4 GHz RF Transceiver.


Sipex. (2002). SP6641A/B - 500mA Alkaline DC/DC Boost Regulator in SOT-23 Rev. 5/26/05.

Texas Instruments. (2008). TPS2375/6/7 IEEE 802.3af PoE POWERED DEVICE CONTROLLERS.
Appendix.

A. Main board schematics as PDF. Separately.
B. Main board PCB image as PDF. Separately.
C. Functional board schematics in PDF. Separately.
D. Functional board PCB image in PDF. Separately.
F. Web page draft in zip. Separately.
G. Bill of materials (BOM).
Bill of materials (BOM) for the Main board.

<table>
<thead>
<tr>
<th>Qty</th>
<th>Value</th>
<th>Part Num</th>
<th>Package</th>
<th>Parts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>B3F-10XX</td>
<td>SW_RST</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>SPC4078</td>
<td>J2</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>L3216C</td>
<td>L1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>P-LCC-2</td>
<td>LED_CD_MRF, LED_CS_SD, LED_EEP</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>1X02</td>
<td>BAT_IN, BAT_OUT</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>1X03</td>
<td>JP1, MRF_CTRL, SD_SW</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>1X04</td>
<td>EEPROM, MRF_SPI</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>1X06</td>
<td>ETHERNET, POWER</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>1X08</td>
<td>SD</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.1uF</td>
<td>C1206</td>
<td>C_EEP, C_TPI, C_TPO</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.1uF</td>
<td>C1206</td>
<td>C_BAT, C_OSC, C_PLL</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.1uF, 16V</td>
<td>C1206</td>
<td>C_BST</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.1uF, 20V</td>
<td>C1206</td>
<td>C_6-7, C_28-29</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1M</td>
<td>R1206</td>
<td>R1, R_MON-</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>IN4004</td>
<td>D041-10</td>
<td>D2, D3, D4, D5</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1uF, ceramic</td>
<td>C1206</td>
<td>C3_S</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2.26k 1%</td>
<td>R1206</td>
<td>R_BIAS</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2V, 10mA</td>
<td>P-LCC-2</td>
<td>LED_PWR</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>8MHz</td>
<td>QS</td>
<td>XTAL1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>10k</td>
<td>R1206</td>
<td>R3_S</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>10k</td>
<td>R1206</td>
<td>R_CD, R_DI_SD, R_EEP_HOLD, R_EEP_WR, R_IRQ_SD, R_RST, R_RST_ETH, R_WP</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>10uF</td>
<td>C1206</td>
<td>C_CAP</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>10uF, 6.3V</td>
<td>C1206</td>
<td>C_CORE</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>10uH</td>
<td>LQH6PP_2424</td>
<td>LQH6PP_S</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>18p</td>
<td>C1206</td>
<td>C_RTC1, C_RTC2</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>18pF</td>
<td>C1206</td>
<td>C_OSC1, C_OSC2, C_OSC_MAIN_1, C_OSC_MAIN_2</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>25AA080SN</td>
<td>SO-08</td>
<td>IC2</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>25MHz</td>
<td>QS</td>
<td>XTAL</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>32.768 kHz</td>
<td>TC38H</td>
<td>CLOCK_OSC</td>
<td></td>
</tr>
<tr>
<td>Quantity</td>
<td>Value</td>
<td>Part Number</td>
<td>Notes</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>--------</td>
<td>-------------</td>
<td>-------</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>49.9kΩ</td>
<td>R1206</td>
<td>R_TPI+, R_TPI-, R_TPO+, R_TPO-</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>68uF</td>
<td>TT5D10</td>
<td>C_OUT</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>68uF</td>
<td>TT5D10</td>
<td>C_IN</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>100μF</td>
<td>C1210</td>
<td>C_IN_S, C_OUT_S</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>100μH</td>
<td>LQH6PP_2424</td>
<td>L_72</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>150Ω</td>
<td>R1206</td>
<td>R_CS_MRF, R_CS_SD, R_EEP_CS, R_PWR</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>330Ω</td>
<td>R1206</td>
<td>R_LEDA, R_LEDB</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>470Ω</td>
<td>R1206</td>
<td>R_RST_I</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>560kΩ</td>
<td>R1206</td>
<td>R_MON+</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>48025-4000-RJ45</td>
<td>48025-4000-RJ45</td>
<td>U$3</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>53375-0610-MINI-LOCK_HEADER</td>
<td>53375-0610-MINI-LOCK_HEADER_6P_ICSP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>BAT-HLD-001</td>
<td>BAT-HLD-001</td>
<td>U$7</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>ENC28J60-SO</td>
<td>SO28W</td>
<td>IC1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>MAX690/802/804/805</td>
<td>SO08</td>
<td>MONITOR</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>MAX5035AUSAB-SO</td>
<td>SO08</td>
<td>72V_DC/DC</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>MBR0520L</td>
<td>SOD123</td>
<td>U$6</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>MBR3100G</td>
<td>D041-7.6</td>
<td>D1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>MRF24J40MA</td>
<td>MRF24J40MA</td>
<td>U$5</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>PIC18F47J13_44PIN</td>
<td>TQFP44_10X10X1MM,2.00MM</td>
<td>U$1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>SCDA7A0200</td>
<td>SCDA7A0200</td>
<td>U$2</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>SP6641</td>
<td>SOT23-5</td>
<td>DC/DC_BAT</td>
<td></td>
</tr>
</tbody>
</table>
## Bill of materials (BOM) for the Functional board.

<table>
<thead>
<tr>
<th>Qty</th>
<th>Value</th>
<th>Part Num</th>
<th>Package</th>
<th>Parts</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td></td>
<td>B3F-10XX</td>
<td>SW_CNNT, SW_FN, SW_RST</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>TC38H</td>
<td>CLOCK_OSC</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>P-LCC-2</td>
<td>LED_CD_MRF, LED_CNNT, LED_EEP, LED_FN</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>SO28W</td>
<td>U$1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>1X02</td>
<td>BAT_IN, BAT_OUT</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>1X03</td>
<td>MRF_CTR</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>1X06</td>
<td>ICSP</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>1X08</td>
<td>PORTA</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>1X14</td>
<td>SIZE_TEMP</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.1u</td>
<td>C1206</td>
<td>C_EEP</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.1uF 20V</td>
<td>C1206</td>
<td>C_PWR</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1uF, ceramic</td>
<td>C1206</td>
<td>C3_S</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>R1206</td>
<td>R3_S</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>10k</td>
<td>M1206</td>
<td>R_CNNT, R_CNNT_LED, R_FN, R_FN_LED, R_RST_P</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>10k</td>
<td>R1206</td>
<td>R_EEP_HOLD, R_EEP_WR</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>10u</td>
<td>C1206</td>
<td>C_TANTAL_CORE</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>10uH</td>
<td>LQH6PP_2424</td>
<td>LQH6PP_S</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>18p</td>
<td>C1206</td>
<td>C_RTC1, C_RTC2</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>25AA080SN</td>
<td>S0-08</td>
<td>IC2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>100uF</td>
<td>C1210</td>
<td>C_IN_S, C_OUT_S</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>150</td>
<td>R1206</td>
<td>R_CS_MRF, R_EEP_CS</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>&lt;=470</td>
<td>M1206</td>
<td>R_RST_I</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>BAT-HLD-001</td>
<td>BAT-HLD-001 U$7</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>MBR0520L</td>
<td>SOD123 U$6</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>MRF24J40MA</td>
<td>MRF24J40MA</td>
<td>U$5</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>SP6641</td>
<td>SOT23-5</td>
<td>DC/DC_BAT</td>
<td></td>
</tr>
</tbody>
</table>
Overview.

1. Eagle editor is file-based. All sheets are the same board.
2. Named nets (with name on it) are connected to each other.
   Nets are connected on all pages.

Sheets.
2. Main MCU.
3. Main MCU, p.2
4. SD card module.
5. MRF24J40MA module.
6. EEPROM with UI-48 module (25AA02E48).
7. Ethernet module (ENC28J60).
8. System main power (DC/DC converter with supervisor).
9. Battery backup circuit (DC/DC converter).
SPI: Slave mode.

Note that SDI/SDO pins must be reversed.

MRF24J40MA module.
Test pins included.
Uses SPI2.

Main_2
2011-05-12 17:11:42
Sheet: 5/9
SPI: Slave mode.

HOLD, WP - disabled.

Note that SDI/SDO pins must be reversed.

EEPROM with UID-48,
Test pins included.
Uses SPI1.
pin 5 NC always leave unconnected.

Decoupling capacitors.

Ethernet connector

Modular Jack w/Integd Magnetics & LED L/F

Ethernet via SPI board.
Test pins included.
Uses SPI.
Diode rectifier, \( V_{out} = (1+R_1/R_2) \times 1.85V \)

Recommended: 6.5V for 3.3V output.

Main power board, DC/DC and supervisor.

C,batt allows battery change while running from Vcc
The current through R1 and R2 should be at least 3mA.

TODO: PFI is connected instead of battery.
Shut down.
Tie this pin to UBATT for normal operation.
Tie this pin the ground to disable all circuitry inside the chip.
Overview.

1. Overview (current page).
2. CPU and peripheral.
3. Power DC/DC converter.
4. MRF radio module.
5. EUI EEPROM module.
Shut down.

Tie this pin to UBATT for normal operation.
Tie this pin to GND to disable all circuitry inside the chip.
SPI: Slave mode.

Note that SDI/SDO pins must be reversed.

MRF24J40MA module.
Test pins included.
Uses SPI2.
SPI: Slave mode.

HOLD, WP - disabled.

Note that SDI/SDO pins must be reversed.

EEPROM with UID-48,
Test pins included.
Uses SPI1.