Project Phantom Heart Pump

Geuvert Hoogkamp
Electrical Engineering – IBE
EWVB4
Project Phantom Heart Pump

Geuvert Hoogkamp
Electrical Engineering – IBE
321132
EWVB4
School Supervisor: Mr J. Zijlstra

Universitair Medisch Centrum Groningen
Hanzeplein 1
9700 RB
Groningen

Company Supervisor: Mr P. M. A. van Ooijen
Foreword

This project started out in October 2011, during the International Biomedical Engineering course of the Hanzehogeschool Groningen. We were a project group of five people with different engineering backgrounds. I would like to thank all of my project members for their contribution:

- Geoffrey Olotu  
  Sensors
- Rutger van der Schoot  
  Mechanics
- Jalte Norder  
  Mechanics
- Thomas Wilken  
  Mechanics

Thomas also helped me during my graduation time and continued with the design and implementations of the PHP during his trainee ship at the UMCG. I would also like to thank Peter M. A. van Ooijen and Volkan Tuncay of the UMCG who commissioned this project and provided funding. Furthermore I would like thank my teacher, Mr. Jan Zijlstra, who supported me during my graduation. I learned a lot of new practical skills in both programming and electronics with this project. Skills I hope I can use in future projects as well.

Geuvert Hoogkamp  
Groningen, 12-06-2012
<table>
<thead>
<tr>
<th>Appendix</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A. Summary design report</td>
<td>37</td>
</tr>
<tr>
<td>B. Summary engineering report</td>
<td>38</td>
</tr>
<tr>
<td>C. Control board schematics</td>
<td>39</td>
</tr>
<tr>
<td>D. Control board lay-out</td>
<td>40</td>
</tr>
<tr>
<td>E. Prototype Schematic</td>
<td>41</td>
</tr>
<tr>
<td>F. Prototype PCB lay-out</td>
<td>42</td>
</tr>
<tr>
<td>G. Development board schematics</td>
<td>43</td>
</tr>
<tr>
<td>H. Development board PCB lay-out</td>
<td>44</td>
</tr>
<tr>
<td>I. D flip-flop calculations</td>
<td>45</td>
</tr>
<tr>
<td>J. D flip-flop schematic</td>
<td>47</td>
</tr>
<tr>
<td>K. Custom dual H-bridge schematic</td>
<td>48</td>
</tr>
<tr>
<td>L. Custom dual H-bridge PCB lay-out</td>
<td>49</td>
</tr>
<tr>
<td>M. Dual H-bridge schematics</td>
<td>50</td>
</tr>
<tr>
<td>N. Dual H-bridge PCB lay-out</td>
<td>51</td>
</tr>
<tr>
<td>O. Display lay-out</td>
<td>52</td>
</tr>
<tr>
<td>P. Mechanical drawings</td>
<td>54</td>
</tr>
<tr>
<td>Q. Afstudeer werkplan Geuvert Hoogkamp EWVB4</td>
<td>63</td>
</tr>
</tbody>
</table>
Summary

The PHP project was commissioned by the radiology department of the UMCG. The main goal of the PHP is to research how diseased aortic valves affect blood flow. To research the effects a system is needed that mimics the behaviour of the heart. For this purpose the PHP will create the different heart rates and stroke volumes. These heart rates are created with a pump that needs to be controlled by electrical hardware and software, which is the main focus of this report. The main question that will be answered in this report is ‘How to realize the hardware and software needed to control the PHP’? To answer this question the choices and the requirements for the hardware and the software will be discussed.

Most of the mechanical parts of the PHP were already available. The only parts that remained were the electronic hardware and software. The hardware, in combination with the software, controls the entire system. At the heart is the microcontroller that controls and monitors the entire system by means of a motor driver and sensors. The motor driver is made up of two full H-bridges and a D-flip-flop circuit that controls the leads of the motor. There are two possibilities for the H-bridge; either a ready-made L6203 IC or a custom H-bridge using MOSFETs. Both options have certain advantages:

L6203 based H-bridge: MOSFET based H-bridge:
- Built-in enable - Current dependant on MOSFET type
- Built-in shoot-through prevention - Voltage dependant on MOSFET type
- One component for a full H-bridge - Heat dissipation dependant on MOSFET type

The D-flip-flop circuit is controlled by a clock signal that is generated by the microcontroller. The sensors are placed inside the system and monitor the pressure and the flow. The data of the sensors is used by the microcontroller for feedback. The parameters for the system are entered by the user via the user interface. The user interface allows the user to monitor the parameters and change them if necessary.

The hardware to control the linear actuator consists of two full H-bridges that are controlled by a dedicated counter made with a D-flip-flop. The full H-bridges are made with the L6203; a ready-made IC. To control the counter and other parts of the system an Atmel At32UC3C1512C microcontroller is used. The software inside the microcontroller controls the hardware and monitors the system with sensors for flow and pressure. The pressure sensor is controlled using the I²C protocol and the flow sensor gives of approximately 2100 pulses per litre. The speed of the linear actuator determines the mimicked heart rate of the system. This heart rate also causes a pressure- and flow- profile of the fluid inside the system. These profiles are monitored with a differential pressure sensor and a flow sensor.

To control the system there is a user interface that is made up of buttons and a character display. The character display provides feedback to the user on the current parameters of the system or shows the current menu position where the user can change the parameters.
## Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC</td>
<td>Analogue to Digital Converter, Converts an analogue voltage to a (digital) numerical value that the software can work with</td>
</tr>
<tr>
<td>CT</td>
<td>Imaging device for the human body that uses X-rays</td>
</tr>
<tr>
<td>IBE</td>
<td>Specialization course of the Hanzehogeschool Groningen</td>
</tr>
<tr>
<td>I²C</td>
<td>Communication protocol that uses only two wires, also called two wire interface (TWI)</td>
</tr>
<tr>
<td>Interrupt</td>
<td>When certain predetermined conditions are met an interrupt can halt the normal software code and execute code that belongs to the interrupt</td>
</tr>
<tr>
<td>Mil</td>
<td>Imperial measurement unit, (1\text{mil} = 0.0254\text{mm}) or (1\text{mm} = 39.3700787\text{mil})</td>
</tr>
<tr>
<td>MOSFET</td>
<td>Voltage controlled electronic ‘switch’</td>
</tr>
<tr>
<td>MRI</td>
<td>Imaging device for the human body that uses magnetism</td>
</tr>
<tr>
<td>PCB</td>
<td>Printed circuit board, piece of plate material that contains connections for the electronic components that are placed on it</td>
</tr>
<tr>
<td>Shoot-through</td>
<td>Effect seen in H-bridges when the sourcing and sinking components of the same lead are both active, causing a short circuit</td>
</tr>
<tr>
<td>Transition table</td>
<td>Table used to aid in the design of digital circuits with Flip-Flop’s</td>
</tr>
<tr>
<td>UMCG</td>
<td>Hospital / University of Groningen</td>
</tr>
</tbody>
</table>
1. Introduction

The Phantom Heart Pump (PHP) project started out as a school project for the first quarter of IBE (International Biomedical Engineering). It was later extended to the second quarter and finally became a graduation project. The project began as collaboration between several disciplines (mechanical-, sensor- and electrical- engineering). The graduation project was part of the electrical engineering discipline. The mechanical engineering was done by a former project member who did his internship at the UMCG. For background information on the preliminary design and engineering phase please refer back to the reports\textsuperscript{1,2} associated with those quarters. The summaries of both reports can be found in appendix A and B.

The PHP project was commissioned by the radiology department of the UMCG. The main goal of the PHP is to research how diseased aortic valves affect blood flow. Pig- or artificial-valves with different sizes will be placed inside a MRI or CT scanner and scanned at different heart rates and stroke volumes. The outcomes of these scans will then be compared to each other. The PHP will mimic the different heart rates and stroke volumes. For this purpose a piston pump is created which is controlled via a microcontroller. A pressure- and flow-sensor are placed in the system to monitor the desired heart rate and stroke volumes.

Another goal of the PHP is to give a reference to the cardiac output measured by a MRI or CT scanner. They need a controllable system which mimics the heart to compare the data of the scanner to. If it is known what was sent in, it is known what the data of the scanner represents. So basically the PHP doubles as a calibration system.

The pump needs to be controlled by electrical hardware and software, which is the main focus of this report. The main question that will be answered in this report is ‘How to realize the hardware and software needed to control the PHP’? As mentioned before, this project started out as a school project. During this time brainstorm sessions were held to come up with solutions on what the system should look like. This project is a continuation of the outcome of those sessions. Therefore the general idea was already there, it just needed to be worked out and implemented.

To answer the main question the report is divided in four major parts.
- Overview of the system
- The electronics hardware
- The software
- Tests and evaluation

The overview will show a complete set-up and tell something about the components. The electronic components will be discussed in depth in the chapter about the electronic hardware, after which the chapter about the software will show how the hardware is controlled. The chapter about tests and evaluation will give an impression of the obtained results of individual parts of the system. The mechanical drawings associated with the PHP can be found in appendix P.

\textsuperscript{1} Phantom Heart Report_1_Design
\textsuperscript{2} Phantom Heart Report_2_Engineering
2. System overview

The complete system consists of a mixture of mechanics, electronics and sensors. The main parts are listed below, with the field behind it.

1. Control interface  electrical / mechanical  
2. Control PCB electrical  
3. Motor control PCB electrical  
4. Motor electrical / mechanical  
5. Piston pump mechanical  
6. One-way-valves mechanical  
7. Pressure sensor sensor / electrical  
8. Flow sensor sensor / electrical  
9. Valve box mechanical  
10. Bypass mechanical  
11. Tubing mechanical  
12. Software electrical (in microcontroller on main control board)

To illustrate the combination of these parts a schematic overview is shown in Figure 2.1. A functional description of the system follows after.

Figure 2.1: System overview
The goal of PHP is to mimic the pumping behaviour of the heart (in pressure, volume and flow). For this purpose a pump is needed that can change speed and volume. To achieve this, a linear actuator (motor) is used which, in turn, drives a piston pump. The motor has a minimum step size \((h)\) of 0.025 \((\pm 0.00)\) mm, together with a piston with a radius \((r)\) of 29.94 \((\pm 0.01)\) mm this gives a volumetric displacement \((V)\) of:

\[
V = \pi \cdot r^2 \cdot h = \pi \cdot 29.950^2 \cdot 0.025 = 70.450mm^3 \approx 0.070ml
\]

The accuracy \((\Delta V)\) is then:

\[
\Delta V = \left| \frac{dV}{dr} \cdot \Delta r \right| + \left| \frac{dV}{dh} \cdot \Delta h \right| = 2 \cdot \pi \cdot r \cdot \Delta r + \pi \cdot r^2 \cdot \Delta h = 2 \cdot \pi \cdot 29.94 \cdot 0.01 + \pi \cdot 29.94^2 \cdot 0 = 1.881mm^3 \approx 0.001ml
\]

Parameters to control the system are changed via the control interface which is directly connected to the control PCB. Parameters which can be changed by the user are:
- Heart rate
- Stroke volume
- Pressure

Two of these parameters can be set by the user and the third value is always a direct result of the two set parameters.

The system functions as follows:
After the parameters are set and a start command is given the pump will pump fluid through the system. The in- and out-lets of the pump are fitted with one way valves to ensure unidirectional flow. When the piston retracts, a negative pressure is created inside the pump, because of the one way valves, until the inlet valve opens. The negative pressure causes fluid to be sucked in through the inlet valve. When the piston is extending the pressure rises inside the pump. When the pressure exceeds the maximum pressure the outlet valve can hold, the fluid is pumped out again. The outlet valve itself opens at 22.5 mmHg. When the system is filled with fluid, the pump has to overcome the pressure inside the system plus the 22.5 mmHg of the valve.
The fluid is led through a flow sensor by means of a bypass to minimize resistance in the overall system as depicted in Figure 2.2. The incoming flow Q1 is split in Q2 and Q3. If flow Q3 is known, the total flow Q1 can be calculated. After the flow sensor the tubing is fed through a hole in the wall into the MRI or CT room. From here on metal parts (or other substances that can cause magnetic interference) are not allowed. The only parts of the system inside the MRI or CT room are the valve holder and its bypass. The valve holder can contain the pig valve or another type of artificial valve for research purposes. The bypass of the valve holder is to redirect flow when the valve needs to be replaced for example. Two manual valves are used to determine the path of flow. After the valve holder the fluid is brought straight back to the pump.

At the in- and out-let of the pump, the leads of a differential pressure sensor are placed to measure the pressure over the valve holder, as can be seen in Figure 2.1.

Figure 2.2: Flow sensor bypass
The data gathered by the sensors is fed back into the control PCB and read by the software. The software uses the data as reference and for feedback purposes towards the user. The values that are shown on the display are values measured by the sensors, not what the microcontroller ‘believes’ is happening. The microcontroller can calculate the heart rate because it sends out the control signal. If the calculated value would be shown on the display, it could occur that a heart rate is shown while the motor is not running. To clarify the software an overall block diagram of the software is shown in Figure 2.3. The main function performs the basic tasks that are not time important and it performs a read operation on the ADC and the sensors when the sensor timer is not active. The motor-timer controls the motor and the sensor-timer performs the read operations of the sensors. The pulse-input-interrupt actually measures the flow speed. The sensor timer reads out the flow speed to have a value at the same time as the pressure. For more information please refer to chapter 4.

![Figure 2.3: Software block diagram](image)
All components are controlled either directly or indirectly by the microcontroller by means of interfaces or peripherals. Some of these peripherals are imbedded in the microcontroller, like the ADC and a timer. Figure 2.4 shows the interfaces and peripherals around the microcontrollers. The arrows indicate data flow direction. The electronic hardware will be treated more in depth in chapter 3.

Figure 2.4: Hardware block diagram
3. Electronic hardware

The electronic hardware (for the remainder of this chapter referred to as ‘the hardware’), has gone through several development stages:

1. Prototype PCB
2. Development PCB
3. Final PCB

The prototype PCB started out as a single PCB with all the components on it. It was a double layer prototype board that was manually etched with the facilities available at the university. Because of the high accuracy needed to create this board, which is difficult to attain at the university, the final PCB (schematics and lay-out can be found in appendix C and D) was ordered at a prototyping service. To be able to continue with the software development in the meantime, a simple development board was made at the university. The schematics and PCB lay-out for this board can be found in the appendixes G and H.

In the final PCB the motor control board is separated from the main control board. The accuracy of the traces needed on the motor control board is lower than on the main control board and is therefore manufactured at the university itself. The smallest clearance that needs to be attained on the main control board, between two copper areas, is 0.2mm. The smallest required clearance on the motor control board, between two copper areas, is 0.5mm.

3.1. Main control board

The main control board could be seen as the heart of the hardware. It contains the following components:

- Microcontroller
- Power supply
- Dedicated counter (for motor control)
- Sensor interfaces
- Control interfaces (buttons and display)
3.1.1. Microcontroller

The choice for the microcontroller was dependant on several minimal requirements. Besides the minimal requirements the microcontroller needs to allow for expansions of the initial requirements in case of upgrades or changes in the design. The minimal requirements are based on the outcome of the brainstorm sessions. Table 3.1 shows a list of the requirements and the outcome.

<table>
<thead>
<tr>
<th>Description</th>
<th>Purpose</th>
<th>Required minimum</th>
<th>AT32UC3C1512C</th>
</tr>
</thead>
<tbody>
<tr>
<td>General Purpose I/O's</td>
<td>Button interface, LCD, motor control</td>
<td>38</td>
<td>81</td>
</tr>
<tr>
<td>USB</td>
<td>Future implementations</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>I²C</td>
<td>Peripheral communication</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>SPI</td>
<td>Peripheral communication</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>Counter / timers</td>
<td>Timer interrupts, pulse counting</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>ADC</td>
<td>Read supply voltages, Sensor output</td>
<td>5</td>
<td>8</td>
</tr>
</tbody>
</table>
| Speed                | High accuracy on timer interrupts | > 20MHz          | 60            
|                      |                                   |                  | At 20 MHz crystal |

Table 3.1: microcontroller requirements

The communication and ADC requirements are chosen so that any type of sensor interface is supported. This also allows for more peripherals in case this is needed in the future. The speed is very important for the timer interrupts. In case of the AT32UCC1512C the fastest clock that can be used for a timer is the base clock divided by two. The highest accuracy that can be attained at a base clock of 60 MHz clock is:

\[
\frac{1}{\text{base clock}} = \frac{1}{30,000,000} = 33.33\text{ns}
\]

This accuracy is needed to precisely control the clock signal that drives the motor. The higher the accuracy the more precisely a certain heartbeat can be simulated. Other parts of the microcontroller that played a role in the decision making was the 32-bit core, which allows large numerical values and the availability of a floating point unit (FPU) which gives the possibility to use decimal numbers in, for example, divisions. Another large influence for this choice was the large online community and support by Atmel itself.

The tasks performed by the microcontroller are:
- Generating a clock pulse for the dedicated counter
- Reading the power supply voltage for the pressure sensor
- Reading the power supply voltage for the motor
- Reading the input generated by the buttons of the control interface
- Generating output for the display of the control interface
- Reading the sensor data

---

1 Phantom Heart Report_1_ Design
For testing and debugging purposes two LED’s are connected to the microcontroller labelled ‘LED1’ and ‘LED2’. For future developments there is also a three pin connection for the USB connected to the Data+, Data- and ground. There is an approximate difference in length of the Data+ trace and the Data- trace of 68mil (1.7mm), where the maximum is 150mil\(^1\) (3.8mm).

The ADC of the microcontroller can measure voltages up to 3V with a 12-bit resolution. The conversion time, at a 12-bit resolution, is near 5.3µs\(^2\). The accuracy of the ADC at a resolution of 12-bit is:

\[
\frac{3V}{2^{12}} = \frac{3000mV}{4096} = 0.73mV
\]

The ADC is used to measure two power supply voltages:
- 8V Power supply for the pressure sensor
- 24 – 48V Power supply voltage of the motor

The other supply voltages are not measured because they are needed by the microcontroller to function. Chapter 3.1.2 gives more information about the supply voltages.

Because the ADC cannot measure voltages above 3V, voltage dividers are used. The supply voltage of the pressure sensor is brought down to approximately 2.966V. The supply voltage of the motor is brought down to approximately 2.604V at a supply voltage of 48V. When the supply voltage of the motor is 24V it will be brought down to approximately 1.302V.

### 3.1.2. Power supply

There are three voltage regulators on the main control board which share a single main power supply. The main power supply should always be greater than 10.5V (the highest output voltage (8V) of the voltage regulators plus 2.5V) and should typically be 12V. The voltage regulators for the ADC reference and the pressure sensor do not need to supply a large current and therefore are not in need of heat sinks or other means of cooling. With the current set-up the voltage regulator for the microcontroller and flow sensor also does not need to be cooled. If the set-up changes in the future, cooling can become necessary, depending on the amount of current needed. For this purpose space is left free behind the 5V regulator to accommodate a heat sink. The power dissipation across the voltage regulator can be calculated by multiplying the main supply voltage minus the output voltage of the regulator with the total current the regulator needs to supply. For a 12V main power supply the following absolute maximum power values can be calculated:

- Pressure supply \((12 – 8) \cdot 0.005 = 0.02W = 20mW\)
- Microcontroller supply \((12 – 5) \cdot 0.5 = 3.5W\)
- ADC supply \((12 – 3.3) \cdot 0.000512 = 0.0045W = 4.45mW\)

---

\(^1\) http://www.intel.com/technology/usb/download/usb2dg_R1_0.pdf (page 7 §3.4)

Electronic hardware
The power calculation for the pressure supply is based on a current consumption of 5mA\(^1\) and the ADC current is based on the specifications in the datasheet\(^2\).

The power calculation of the microcontroller supply is an estimation based on the maximum current the I/O’s can source\(^2\) and the consumption of the flow sensor which is a maximum of 10mA\(^3\). The microcontroller can source an absolute maximum of 480mA and uses approximately 40mA when active\(^2\). The ADC power supply is merely a reference voltage and does not need to source any current. For reference an estimation of 512\(\mu\)A is used based on an active current usage of 512\(\mu\)A/MHz by the microcontroller and a 1 MHz clock of the ADC\(^2\).

### 3.1.3. Dedicated counter

This paragraph is mostly about the hardware that provides the step sequence. For information about the motor control board see chapter 3.2. The motor is a linear actuator that needs to be controlled like a stepper motor. This means an alternating current through the two coils of the motor. Figure 3.1\(^4\) shows the step sequence needed to turn the motor clockwise or counter clockwise.

![Figure 3.1: Motor step sequence](image_url)

There are two straightforward ways to create this sequence. The first is to control each lead of the motor directly from the microcontroller. The second is to use a dedicated counter (D-flip-flop circuit) to control the leads of the motor. The advantage of the counter is that the microcontroller only needs to create a clock signal to put the counter in the next state, a direction signal to set the direction and an enable signal. The disadvantage is that you cannot deviate from the sequence set at forehand. Reasons to deviate are:

- Preventing shoot-through (in the H-bridge, see chapter 3.2)
- High currents at low frequencies

---


\(^3\) [http://docs-europe.electrocomponents.com/webdocs/064e/0900766b8064e57e.pdf](http://docs-europe.electrocomponents.com/webdocs/064e/0900766b8064e57e.pdf)

By using time delays in the circuit, shoot-troughs can be easily prevented and high currents can be prevented using the enable. If the leads were directly controlled by the microcontroller, the microcontroller would need to create the delay between the switching on of the FETs (see chapter 3.2), complicating the software the more than necessary. Taking this into account and the fact that the software becomes much simpler, the dedicated counter is used.

The counter circuit is made up of a D-flip-flop, AND-gates, OR-gates and an inverter. To determine what the circuit should look like, a transition table was created which can be found in appendix I. The outcome of the transition table was as following:

\[
\begin{align*}
D_3 &= CW \times Q_1' + CW' \times Q_0' \quad (X' \text{ indicates a logical NOT}) \\
D_2 &= CW \times Q_1 + CW' \times Q_0 \\
D_1 &= CW \times Q_2' + CW' \times Q_3' \\
D_0 &= CW \times Q_2 + CW' \times Q_3
\end{align*}
\]

D₃ to D₀ correspond respectively to A, A\,\backslash, B and B\,\backslash. The CW signal is the direction signal. Appendix J shows the circuit diagram of the counter. To be able to set the direction, the CW signal is connected to four AND-gates directly, and to four AND-gates via an inverter. This way you can select which outputs are connected to which inputs of the D-flip-flop to create the desired order (clockwise or counter clockwise) of the sequence shown in Figure 3.1. The OR-gates are there to prevent the outputs of the AND-gates to be connected to each other.
3.1.4. Interfaces

The interfaces are the connections on the control board that allow interfacing with different parts of the system. The control board cannot function properly without these connections. These interfaces can be found on the edges of the board, as depicted in Figure 3.2.

1. Main power in, typically 12V
   *Main power connection of the control board. Positive pin indicated with ‘12V’.*

2. Flow sensor
   *The flow sensor gives a pulse output which is connected to the counter input of the microcontroller. Approximately 2100 pulses per litre. Accuracy: ±1.5%.*

3. Pressure sensor
   *Pressure sensor is controlled via I²C. Accuracy: ±0.2%.*

4. JTAG
   *Program and debug interface for the microcontroller.*

5. LCD control interface
   *Parallel communication interface to control the display.*

6. Button connections
   *See below.*

7. Motor control board connection, lead connection, enable and Vcc
   *See below.*

8. USB, for future purposes the USB pins of the microcontroller are connected to these
   *See below.*

Figure 3.2: Interface connections

1. Main power in, typically 12V
2. Flow sensor
3. Pressure sensor
4. JTAG
5. LCD control interface
6. Button connections
7. Motor control board connection, lead connection, enable and Vcc
8. USB, for future purposes the USB pins of the microcontroller are connected to these
All the button connections are connected to inputs on the microcontroller. There is one common 5V line which is connected to all the buttons. When a button is pressed the input on the microcontroller will be driven to 5V. One special connection is the key switch (see Figure 3.3). The enable signal is fed through the key switch so that the operator can prevent the motor from switching on. The signal that goes from the key switch to the motor enable is also brought back to an input of the microcontroller for feedback purposes.

![Figure 3.3: Keyswitch principle](image)

The interface with the motor control board contains several signals:
- A ground for equipotential
- The four signals to control the steps (D3 to D0)
- The enable signal
- The Vcc of the motor

The four signals that control the steps of the motor are connected to the inputs of the H-bridges. The signals control directly which lead of the motor is connected to the Vcc and which is connected to ground. The Vcc for the motor itself comes from a separate power supply; it does not originate from the control board. The Vcc of the motor is connected to the ADC of the microcontroller via a voltage divider. By connecting the Vcc the microcontroller can monitor the supply voltage of the motor.

For future implementations the USB pins (Data+ and Data-) of the microcontroller are connected to a pinheader, together with the ground. The maximum deviation in length of the traces that carry the data signals is taken into account during tracing. The maximum deviation may be up to 150mil \(^1\) (3.8mm); the obtained deviation is 68mil (1.7mm). In case USB implementations are desired in the future it is advised to create a second PCB to which the Vcc of the control board is connected together with the connection of the pinheader. The necessary components can then be placed on this PCB according to the USB industry standards.

---

\(^1\) http://www.intel.com/technology/usb/download/usb2dg_R1_0.pdf (page 7 §3.4)
3.2. **Motor control board**

The motor control board consists of two full H-bridges, buffer capacitors and fly back diodes. Appendix K to N show the schematic and lay-out of two motor control boards, both showing the components mentioned previously. The buffer capacitors buffer energy to drive current to and from the motor. They store the energy released by the coils of the motor, when the current direction is changed, to minimize peak voltages and buffer the current when the direction of the motor changes to minimize the strain on the power supply. The fly back diodes serve a similar purpose; they allow the current to flow away from the coils of the motor to minimize peak voltages. A diode is also placed after the inlet of the power supply to prevent a current from going back in to the power supply.

There are two options for the motor control board:
- L6203 (DMOS full bridge driver) based circuit board
- Custom MOSFET based H-bridge circuit board

The L6203 has the following specifications:\(^1\):
- Nominal current of up to 4A
- Peak current up to 10A
- Voltage of up to 48V

The motor supply voltage is near the limits of the L6203 when the motor is used at its maximum operating voltage of 48V. The diode near the main inlet lowers the voltage with 0.6V and thus keeping everything within the normal limits. Under normal circumstances the power supply is 24V, instead of 48V.

The advantages to using the L6203 are:
- Built-in enable
- Built-in shoot-through prevention
- One component for a full H-bridge

The built in shoot-through prevention makes the need for individual control of the motor leads by the microcontroller obsolete. When the motor is driven at frequencies lower than 1 kHz the current can become very high. The enable gives the possibility to stop conduction of the H-bridge and thus preventing too high a current from flowing. The fact that the entire H-bridge is in a single housing also makes the design of the lay-out far less complicated.

---

To calculate the current at a given frequency, the period time of that signal \( t \), the resistance \( R \) of one coil of the motor, the inductance \( L \) of that coil and our operating voltage \( E \) are needed. According to the datasheet the resistance of one coil is 1.75Ω and the inductance is 3.3mH.

For an operating speed of 1 kHz at 48V this gives a current of:

\[
i(t) = \frac{E}{R} \cdot \left(1 - e^{-\left(\frac{R}{L}\right) \cdot t}\right) = \frac{48}{1.75} \cdot \left(1 - e^{-\left(\frac{1.75}{3.3 \times 10^{-3}}\right) \cdot 0.001}\right) = 11.29.
\]

At 1.5 kHz the current is already reduced to:

\[
i(t) = \frac{E}{R} \cdot \left(1 - e^{-\left(\frac{R}{L}\right) \cdot t}\right) = \frac{48}{1.75} \cdot \left(1 - e^{-\left(\frac{1.75}{3.3 \times 10^{-3}}\right) \cdot 0.00667}\right) = 8.17A.
\]

At a frequency of 1.5 kHz the current is already up to more than 8A. This is only the first step in the ramping of the frequency, after this the current lowers very fast, causing no ill effects to the circuitry. But when the frequency needs to go below 1.5 kHz it becomes vital to control the current.

In a worst case scenario (when all inrush phenomena are gone) the current could become:

\[
i = \frac{E}{R} = \frac{48}{1.75} = 27.43A
\]

Besides the L6302 based circuit, an H-bridge based on MOSFETs is also a possibility. The advantages of a MOSFET based H-bridge are:

- Current dependant on MOSFET type
- Voltage dependant on MOSFET type
- Heat dissipation dependant on MOSFET type

The MOSFET H-bridge does the same as the L6203. All the special features, however, have to be added in another way. To create an enable, an extra MOSFET needs to be used to cut off the current directly after the main power inlet. The shoot-through prevention needs to be created with RC-circuits that create a very small off-to-on delay of 7.26µs at the sinking MOSFETs. This delay must only occur when the MOSFETs are ‘turned on’, so a diode is placed over the RC-circuits to allow for a quick on-to-off transition. Appendix K shows the schematics for the MOSFET based dual full H-bridge circuit. The RC-circuits can be seen at the sinking MOSFETs.

The sourcing MOSFETs are controlled with a special voltage divider. To operate the sourcing MOSFETs, the gate voltage needs to be typically 5V lower than the drain voltage, with a maximum difference of 20V. Because a normal voltage divider (with a division of two) would create a voltage difference of more than 20V, at a supply voltage of 48V, a zener diode of 5.6V is used (a different division could be used here, but a zener diode ensures uniform operation). The zener diode is placed parallel to the top resistor ensuring that the gate is always 5.6V lower than the drain, when activated. The zener diode doubles as a normal diode that allows for a quick on-to-off transition that is not limited by the value of the top resistor. A great disadvantage of the custom H-bridge is the amount of components and space needed on the PCB.
The choice for the MOSFETs is mainly based on availability at the university. Furthermore they needed to be able to switch more than 27.43A and be able to withstand voltage greater than 48V. The two types of MOSFETs that are used:
- P-channel IRF4905
  - -55V
  - -74A
- N-channel IRF1010E
  - 60V
  - 84A

The P-channel MOSFETs switch the coils to the Vcc (called sourcing) and the N-channel MOSFETs switch the coils to ground (called sinking). P-channel MOSFETs are used to prevent the need for driver IC’s that boost the gate voltage above the threshold level, which is above Vcc in case of an N-channel MOSFET.

During the development of the system problems arose with the motor. At lower frequencies of 1500Hz the motor worked just fine. When a slightly higher frequency, of even 2000Hz was used, the motor refused to work properly. According to the manufacturer, Nanotec, this is because the frequency needs to be ramped up. This means that after every step in the sequence (see Figure 3.1) the frequency can be increased. This also means that after every change in direction the ramping needs to start over again.

After a while a second problem arose. When operating the motor at higher frequencies, of more than 7000Hz, the motor started to ‘stutter’. If the shaft of the motor was pushed or pulled during operation there was no problem but when the motor was left alone, it could not be used to its full potential. The problems with the motor were of a non-electrical nature. When the motor is laid loosely on a surface and then operated, it starts to vibrate violently. These vibrations influence the motor in a disastrous way and occur at frequencies above 7000Hz. When the motor is fixated, by pressing it firmly on a stiff surface or mounting it on the pump cylinder, the vibrations are reduced significantly and thus allowing for normal operation of the motor.

In the end the motor control board based on the L6203 is used for the following reasons:
- Simpler design lay-out
- Fewer components
- Smaller board size
- Built-in shoot-through prevention
- Built-in enable signal

The schematic and PCB design of the motor control board with the L6203 can be found in appendix M and N. The schematic and lay-out of the MOSFET based motor control board are in appendix K and L.
4. Software

The software is the code that is executed inside the microcontroller to control the system. It monitors the sensors and acts accordingly to the gathered data from the sensors and the parameters set by the user. The microcontroller used is the AT32UC3C1512C from Atmel. For programming and debugging Atmel provides its own compiler called AVR studio\(^1\). A piece of hardware is required for communication with the microcontroller, a so called programmer. The programmer used is the AVR Dragon; a versatile programmer for a ‘reasonable’ price\(^2\).

The microcontroller can be seen as the heart of the system. It takes care of all of the measuring and actuation tasks needed for a functioning system. The tasks performed by the microcontroller are:

- User I/O
  - Measuring user input
  - Generating display data
- Peripheral I/O
  - Pressure sensor
  - Flow sensor
  - H-bridge control
  - Voltage measuring

4.1. User Interface

The user interface is the part of the system which allows interaction with the user. The user can set certain parameters and read actual values from the system via this interface. The user interface uses by far the most pins of the microcontroller; eleven pins are reserved to control the display and twenty-three of the pins are used for the buttons and the key switch.

The user input can be monitored in three ways:

- Interrupt routines
- Timer-enabled-check
- Main-enabled-check

The use of interrupt routines would allow for the fastest reaction when the user wants to change anything. It would, however, mean an additional twenty-two (the connection to the key switch is an output) interrupt routines to change something relatively insignificant. The chance that an interrupt, generated by pressing one of the buttons, disturbs the timer-enabled-interrupt that controls the counter (see chapter 4.3) also increases significantly.

Another option would be to use another timer-enabled-interrupt, in which a check will be performed to see if any button is pressed anew. This would be rather pointless however, because the use of a timer-enabled-interrupt is only useful in case of a time dependant routine, which leaves the main-enabled-check. This means the check will be performed in the normal ‘main’ of the program, which loops automatically.

---

\(^1\) [http://www.atmel.com/microsite/avr_studio_5/default.aspx](http://www.atmel.com/microsite/avr_studio_5/default.aspx)

The principle of the main-enabled-check is the same as the timer-enabled-check with the exception of the fact that there is not an exact time interval in-between checks and that a check can be interrupted. Since the checking of the user input is not a high priority this is no problem.

Figure 4.1 shows the flowchart for the button check. Every time the button check is called in the main function the software will check whether or not a button is pressed, if it was pressed during the previous check and, when necessary, perform its corresponding action. In most cases the function of the button depends on the menu position the software is in.

Please keep in mind that the flowcharts’ purpose is to give an understanding of what the software does, not how it exactly works.
The Information that is to be shown on the screen is dependent on the menu position and the current values. When the menu is on the ‘home’ position, the screen will look like Figure 4.2 (other screen lay-outs are shown in appendix O). Most information on this screen is static, meaning the text will neither change nor change position. Those static parts need only to be placed when the menu position is set to ‘home’. After this only the dynamic information like the BPM needs be updated. This goes for most of the screens. To do this it is easiest to call a subroutine of the display driver every time a new value is calculated. The part of the software that calculates the BPM then controls the value on the screen that represents the BPM.

Basically every time a character is placed on the display the routine looks the same. Figure 4.3 shows the flowchart on how to place a character. For more information on what the data looks like please refer to the datasheet of the HD44780\(^1\).

![Flowchart of display data](image)

**Figure 4.2: Home screen**

<table>
<thead>
<tr>
<th>Heart rate</th>
<th>080 BPM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pulse volume</td>
<td>070 ml</td>
</tr>
<tr>
<td>Pressure</td>
<td>120 mm Hg</td>
</tr>
<tr>
<td>menu</td>
<td>QS</td>
</tr>
</tbody>
</table>

**Figure 4.3: Display data**

---

4.2. **Measuring sensor data**

The system uses two different types of sensor; a pressure sensor and a flow sensor. The pressure sensor can be read by means of I²C and the flow sensor gives of a pulse signal, the more pulses per time unit, the higher the flow. To communicate with the pressure sensor two dedicated I²C pins on the microcontroller are connected to the pressure sensor interface. Figure 4.4 shows how the data from the pressure sensor is read. The measuring of the sensors is activated by means of a timer. This also allows for future implementations of graphs if this would be desired. This timer also places the value of the flow sensor on the display. The reading of the flow sensor is discussed after this.

![Flowchart](image-url)
The flow sensor is a little less sophisticated. It gives of a number of pulses per time unit, depending on the flow through the sensor. The number of pulses per time unit must be measured by the microcontroller. For this purpose a counter/timer pin of the microcontroller is used. This pin is internally connected to a clock signal for reference. This way the time between each pulse can be measured and thus give an accurate measurement of the flow without too much of a delay. Every time a pulse is measured an interrupt is generated. In this interrupt the time can be read and worked with, as shown in Figure 4.5.
4.3. **H-bridge control**

To control the H-bridge three signals are required.

1. An enable signal
2. A direction signal
3. A clock to put the counter in the next state

As mentioned in chapter 3.2 the frequency of the clock signal needs to be ramped. To do this a timer-enabled-interrupt is used of which the period keeps decreasing. The timer of the AT32UC3C1512C works with a compare register (RC). A timer register is incremented every time an internal clock gives a pulse. When the timer register value matches the RC value an interrupt is generated. This timer keeps on running while the pump should function. Inside the interrupt the position of the pump is being kept track of. When a pre-determined position is reached the direction signal will change and the ramping will start over from the starting frequency. As long as the pump is to remain active the enable signal will be active. Figure 4.6 shows the flowchart of the interrupt that controls the pump. The enable signal is controlled outside of the interrupt in the main part.

![Flowchart of the H-bridge control](image.png)
4.4. **Voltage measuring**

The measuring of the voltage is done by the differential ADC of the microcontroller. This means that the voltage on pin ‘a’ is measured with reference to pin ‘b’. In this case pin ‘b’ is connected to ground. The two voltages measured are the power supply for the pressure sensor and the power supply for the motor. These voltages are measured so that the microcontroller ‘knows’ if the system is able to function normally. These voltages are checked during start up as part of a POST (Power On Self-Test) and during normal operation.

The checking of the power supply for the pressure sensor is quite important to prevent a hang-up of the system. When the pressure sensor is being addressed, the function waits until data is received. If the pressure sensor is not powered the microcontroller will never receive data and stay in an infinite waiting loop.

Figure 4.7 shows the flowchart of the read ADC function.

![Flowchart of Read ADC](image_url)

Software
5. Tests and evaluation

The PHP is a combination of multiple components. Each component is important in its own right, but all components need to work together in order to create a fully functional device. The PHP consists of the following major components:

1. Motor control
   a. Clock signal generation
   b. Enable control
   c. Direction control

2. Peripheral feedback
   a. Pressure sensor
   b. Flow sensor
   c. ADC check

3. User input and output control
   a. Visual feedback via LCD
   b. User control via buttons

At the time of writing only the motor control and the visual feedback are operational.

5.1. Motor control

The clock signal is needed to put the counter in the next state. The counter, in turn, controls the leads of the motor. The timing (or the frequency) of the clock signal determines the speed of the motor and with it the current that flows through the coils of the motor. At high frequencies (1500Hz and above) the current will remain relatively low, but at frequencies below 1500Hz it becomes imperative to use the enable signal to switch off the H-bridge and thus limiting the current. The enable signal is also used to prevent accidental operation of the motor and is only active when the motor needs to operate. The direction is controlled by the same timer-interrupt that controls the clock signal. When the motor reached its predetermined position the direction is switch around.

The clock signal is one of the more vital sub-components of the PHP. It determines the mimicked BPM and the pressure- and the flow- profile of the fluid inside the system. Figure 5.1 shows the frequency progress of the clock signal versus the step number (the shaft of the motor can be extended in 2000 steps). The frequency progress is almost linear; it is not fully linear due to the inaccuracy of the timer. This progress is attained by incrementing the frequency almost evenly across all the steps and is the most basic method to create the clock signal. It also limits the BPM that can be attained.

To attained a certain pulse volume there are so many steps available. Within these steps the frequency needs to be incremented enough to reach a certain BPM. When the progress of the frequency is linear (as is the case in Figure 5.1), the BPM that can be reached is limited by the time needed to reach the maximum frequency. To be able to reach a higher BPM the frequency progress needs to be different.
The frequency is inversely proportional to the period. A graph of the period of the frequency from Figure 5.1 is shown in Figure 5.2. The area underneath the graph line represents the time needed for the piston to go in one direction; this means that the area times two represents the time for one ‘heartbeat’. By decreasing the area underneath the graph line it is possible to increase the maximum BPM that can be attained.
Figure 5.3 shows a possibility to decrease the area underneath the graph line. By ramping down the period, or ramping up the frequency in the first five hundred steps the total time is decreased significantly. The speed with which the frequency can be increased each step, depends on the frequency that needs to be bridged. If the increase in frequency needed is relatively low, fewer steps are needed than when a larger frequency needs to be bridged. This is caused by the time needed to build-up the magnetic field inside the motor. Actual values for the magnetic field needed are not given by the manufacturer. This means that the speed, with which to increase the frequency, needs to be found out by trial and error.

![Figure 5.3: Period progress vs. step](image)

### 5.2. User input and output control

User input is generated by means of buttons. These buttons allow the user to navigate through the menu, alter settings and values and start and stop the system. The user input is not implemented in the system at the time of writing yet and could therefore not be tested.

User output is created with a display that gives feedback to the user on what the current parameters are or what the current menu position is. The display is a character display that is made up of four lines that can contain 20 characters each. Figure 5.4 shows an example of the home screen. This is the default screen which shows the current heart rate, pulse volume and pressure. At the bottom the function of the menu buttons are shown. In case of the home screen these are the ‘menu’ button and the ‘QS’ (Quick Setup) button. Appendix O shows more schematic examples of user output on the display.
The display contains a HD44780 controller chip from Hitachi\(^1\). This is a very common controller used on many types of character displays and a lot of information about this controller can be found on the internet\(^2\). Once the display is powered it needs to be initialized before it can be used. The basic initialization sequence looks like this:

1. Write 30h, three times
2. Set interface length (4- or 8-bit)
3. Enable display / cursor
4. Clear screen and move cursor ‘home’
5. Set cursor direction

The display for the PHP is set to an 8-bit interface, cursor off and cursor direction from left to right. The cursor is turned on when it is necessary to clarify the position of the cursor; for example when entering a value.

---

\(^1\) [http://www.adafruit.com/datasheets/HD44780.pdf](http://www.adafruit.com/datasheets/HD44780.pdf)

\(^2\) [http://joshuagalloway.com/lcd.html](http://joshuagalloway.com/lcd.html)
6. Conclusion

The PHP contains a piston pump that pumps fluid through a tubing system. This tubing system is connected to a valve holder which contains a pig valve, or an artificial valve, for research purposes. The main goal of this report was to answer the question ‘How to realize the hardware and software needed to control the PHP’?

The piston pump of the PHP is driven by a linear actuator that needs to be controlled by hardware and software. There are two coils in the actuator that need an alternating current to provide linear movement. This alternating current is created with two full H-bridges that are controlled by a dedicated counter made with a D-flip-flop. The full H-bridges are made with a ready-made IC; the L6203. To control the counter and other parts of the system the Atmel AT32UC3C1512C microcontroller is used. The software inside the microcontroller creates a clock signal that controls the dedicated counter. By changing the period of the clock signal the speed of the linear actuator can be controlled.

The speed of the linear actuator determines the mimicked heart rate of the system. This heart rate also causes a pressure- and flow- profile of the fluid inside the system. These profiles are monitored with a differential pressure sensor and a flow sensor. The sensors are read by the microcontroller that uses the data for feedback. The pressure sensor is controlled using the I²C protocol and the flow sensor gives of approximately 2100 pulses per litre. Besides the pressure- and flow- sensor there is also an ADC to monitor the power supply voltages of the pressure sensor and the linear actuator.

To control the system there is a user interface that is made up of buttons and a character display. The character display provides feedback to the user on the current parameters of the system or shows the current menu position where the user can change the parameters by means of the buttons.
Citations

Reports:
- Phantom Heart Report_1_Design
- Phantom Heart Report_2_Engineering

Webpages:
- http://joshuagalloway.com/lcd.html

PDF documents:
- http://docs-europe.electrocomponents.com/webdocs/064e/0900766b8064e57e.pdf
Appendix

A  Summary design report
B  Summary engineering report
C  Control board schematics
D  Control board lay-out
E  Prototype Schematic
F  Prototype PCB lay-out
G  Development board schematics
H  Development board PCB lay-out
I  D flip-flop calculations
J  D flip-flop schematic
K  Custom dual H-bridge schematic
L  Custom dual H-bridge PCB lay-out
M  Dual H-bridge schematics
N  Dual H-bridge PCB lay-out
O  Display lay-out
P  Mechanical drawings
Q  Afstudeer werkplan Geuvert Hoogkamp EWVB4
A. Summary design report

In this project, we were given a task by the UMCG (University Medical Centre Groningen) to design a Heart Phantom Pump device that is compatible to MRI/CT scanners for heart analyses. The device we were asked to design should be able to carry out flow and pressure measurement as well as being able to determine the stroke volume and heart rate through the system. The concept of the Heart Phantom Pump is a heart pumping device that mimics the behaviour of the heart. In other words, the device act as a heart pumping system that creates a pulsatile flow of blood which flow and pressure generated in the system can be detected and measured with proper monitoring systems.

The first step we took in analysing the project task was to have a meeting with our project customer (representative from UMCG). In the meeting, we discussed various possibilities of designing the Heart Phantom pump device and after carefully analysing these possibilities; it was clear what we needed to design for our customer. Furthermore, we carried out series of brainstorming sessions in order to determine the best possible design for the Heart Phantom Pump, as well as the requirements for the Heart Phantom pump device. This led us to breaking down the various parts of our system design into components according to how we envisage our final prototype. The components that were determined to be the main parts of our Heart Phantom pump design include; a pumping device, sensors, tubing system, valve system, electronics and a control interface. These components were analysed in terms of their requirements and function to our system design. By carefully analysing each of these components, we were able to select the best method of designing each of these components.

The pumping device as the name depict act as a blood pumping device for our system and it is the component that ensures a pulsatile flow would be generated in the Heart Phantom pump and transported to every other parts of the system. The next component that was analysed in our system design was the sensors. As earlier explained, the goal of this project was to measure pressure and flow in the system in order to determine the heart-beat and stroke volume. With this in mind, it was important for us to come up with a nice way to measure flow and pressure in the system. Therefore, we choose two ultrasonic sensors that will be able to carry out pressure and flow measurement in our system without obstructing the flow generated by the pumping device. Next to measuring pressure and flow measurement was the tubing system which is made up of plastic material and will be used in the circulation of blood around the system. The valve was another component that was seen to be vital in our Heart Phantom Pump design as it would be acting like the valve in a real heart and will be used in controlling flow in the Heart Phantom system. Finally, we have the control interface and electronics which would be used to display information measured by the sensors as well as simulating the flow in the pumping device. The control interface will be displaying information about the heart-rate and stroke volume and will be acting as the central control unit for our Heart Phantom Pump device. With all these various components interacting with each other, we will be able ensure that proper flow and pressure measurement would be carried out in our system and we will be able to design and simulate an artificial heart system for our customer.
B. Summary engineering report

In the final phase of the project Phantom Heart, we have been able to implement the design we made in the first quarter of the project. During the design phase, we selected all the required components needed to create the Phantom Heart system. We further made a complete assembly of how we envisioned the Phantom Heart system to function. With this in mind, we were able to move on to the next phase, which was the implementation phase. In the implementation phase, we altered our design by changing our system from an open system to a closed system which further affected the sensors selected for pressure and flow measurements. The sensor principles of measurement were optimised from ultrasonic flow and pressure principle of measurement to differential and Hall Effect principle of pressure and flow sensing. After making these changes, we were able to move on to the next step of building our system.

The implementation of our system began with ordering of various parts after proper research had been carried out. Various aspects such as accuracy, cost, reliability, range of measurement, electrical connection, medium of measurement were all carefully invested prior to making a decision on ordering components for our design. With the ordering and arrival of components, we were able to start up with the assembly of our system. We divided the implementation phase into two parts namely: “Mechanical construction and Electrical interface. The mechanical part which is made up of stroke pump and a linear actuator that is able to generate pulse stroke at a speed of 250 mm/sec. The pump makes connection to a tubing system which the liquid flow generated by the pump can be channelled through the valve system. The valve system is simply a T-shaped valve with a holder. The valve has a bypass which liquid can flow through in different directions thereby making the liquid flow through the valve very efficient. Furthermore, we have two sensors (pressure and flow) that are able to detect flow and pressure in the Phantom Heart system. The flow sensor “Fluid Flow sensor” measures liquid flow at a rate of 0.25 to 6.5 L/min and pressure sensor “DRMOD-I2C” measures the system’s pressure at a pressure range of 0 – 1 bar. These sensors were connected to the tubing system where flow and pressure measurement were carried out by both sensors.

The final implementation of our design was the electronics which controls the entire system functioning. Both components (pump and sensor) were connected to a microcontroller where they were both powered. With this connection, the electronics was used to control the pump stroke to initiate liquid flow to the system at the required speed, and sensor signal was directly sent to the microcontroller where flow and pressure values were displayed on the main board of the electronics. With the interface of electronics to the pump and sensors, we were able to perform test with the Phantom Heart system and the performance of the system were carefully monitored. In the end, we were able to combine our knowledge of mechanical engineering, electrical engineering and sensor technology in the design of a system that uses pump, sensors, tubing and valve to generate flow circulation with a stroke volume that mimic the circulation behaviour of the left ventricle.
C. Control board schematics
D. Control board lay-out
E. Prototype Schematic
F. Prototype PCB lay-out
G. Development board schematics
H. Development board PCB lay-out
## I. D flip-flop calculations

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A'</th>
<th>B'</th>
<th>A</th>
<th>A'</th>
<th>B</th>
<th>B'</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Top to bottom: CW  
Bottom to top: CCW

### Clockwise

<table>
<thead>
<tr>
<th>Q₃</th>
<th>Q₂</th>
<th>Q₁</th>
<th>Q₀</th>
<th>D₃</th>
<th>D₂</th>
<th>D₁</th>
<th>D₀</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

### Counter clockwise

<table>
<thead>
<tr>
<th>Q₃</th>
<th>Q₂</th>
<th>Q₁</th>
<th>Q₀</th>
<th>D₃</th>
<th>D₂</th>
<th>D₁</th>
<th>D₀</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

\[
D₃ = CW \times Q₁' + CW' \times Q₀'
\]

\[
D₂ = CW \times Q₁ + CW' \times Q₀
\]

\[
D₁ = CW \times Q₂' + CW' \times Q₃'
\]

\[
D₀ = CW \times Q₂ + CW' \times Q₃
\]
Phantom Heart Pump project

Clockwise

<table>
<thead>
<tr>
<th>D₃</th>
<th>Q₃ Q₂</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>1</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>01</td>
<td>x</td>
<td>1</td>
<td>x</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>10</td>
<td>x</td>
<td>1</td>
<td>x</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>D₂</th>
<th>Q₃ Q₂</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>01</td>
<td>x</td>
<td>0</td>
<td>x</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>10</td>
<td>x</td>
<td>0</td>
<td>x</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>D₁</th>
<th>Q₃ Q₂</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>1</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>01</td>
<td>x</td>
<td>1</td>
<td>x</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>10</td>
<td>x</td>
<td>1</td>
<td>x</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>D₀</th>
<th>Q₃ Q₂</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>01</td>
<td>x</td>
<td>1</td>
<td>x</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>10</td>
<td>x</td>
<td>0</td>
<td>x</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Counter clockwise

<table>
<thead>
<tr>
<th>D₃</th>
<th>Q₃ Q₂</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>1</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>01</td>
<td>x</td>
<td>0</td>
<td>x</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>10</td>
<td>x</td>
<td>0</td>
<td>x</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>D₂</th>
<th>Q₃ Q₂</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>01</td>
<td>x</td>
<td>1</td>
<td>x</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>10</td>
<td>x</td>
<td>1</td>
<td>x</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>D₁</th>
<th>Q₃ Q₂</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>1</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>01</td>
<td>x</td>
<td>1</td>
<td>x</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>10</td>
<td>x</td>
<td>1</td>
<td>x</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>D₀</th>
<th>Q₃ Q₂</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>01</td>
<td>x</td>
<td>1</td>
<td>x</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>10</td>
<td>x</td>
<td>1</td>
<td>x</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
J. D flip-flop schematic

Dedicated Counter D Flip-Flop

Phantom Heart Pump project

12-Jun-12
K. Custom dual H-bridge schematic
L. Custom dual H-bridge PCB lay-out
**M. Dual H-bridge schematics**

![Dual H-bridge schematics diagram]

- 24 - 48V supply in
- Dual H-bridge schematics
- Key components: IC1, IC2, C1, C2, C3, C4, D1, D2, Q1, Q2, Q3, Q4, SW, JP1, JP2, SW1, SW2, SW3, SW4, SW5
- Pump connection: X1-1, X1-2, X1-3, X1-4
N. Dual H-bridge PCB lay-out
### O. Display lay-out

<table>
<thead>
<tr>
<th>Heart rate</th>
<th>080 BPM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pulse volume</td>
<td>070 mL</td>
</tr>
<tr>
<td>Pressure</td>
<td>120 mmHg</td>
</tr>
<tr>
<td>menu</td>
<td>QS</td>
</tr>
</tbody>
</table>

Figure 0.1: Menu main screen

<table>
<thead>
<tr>
<th>Set Heart rate</th>
<th>080</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use number pad to change</td>
<td></td>
</tr>
<tr>
<td>Done to save &amp; exit</td>
<td></td>
</tr>
<tr>
<td>▼</td>
<td>^</td>
</tr>
</tbody>
</table>

Figure 0.2: Menu set heart rate

<table>
<thead>
<tr>
<th>Set Pulse volume</th>
<th>070</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use number pad to change</td>
<td></td>
</tr>
<tr>
<td>Done to save &amp; exit</td>
<td></td>
</tr>
<tr>
<td>▼</td>
<td>^</td>
</tr>
</tbody>
</table>

Figure 0.3: Menu set pulse volume

<table>
<thead>
<tr>
<th>Set Pressure</th>
<th>120</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use number pad to change</td>
<td></td>
</tr>
<tr>
<td>Done to save &amp; exit</td>
<td></td>
</tr>
<tr>
<td>▼</td>
<td>^</td>
</tr>
</tbody>
</table>

Figure 0.4: Menu set pressure
System settings
Change settings?
OK back

Figure 0.5: Menu system settings

Quick setup

HR 80
PV 70
Press 120
\^ \^ next done

Figure 0.6: Menu quick setup
P. Mechanical drawings
Overzicht ISO (1:2)
Q. Afstudeer werkplan Geuvert Hoogkamp EWVB4

Competenties

**Beroepscapaciteiten:**

<table>
<thead>
<tr>
<th>Kenmerk:</th>
<th>De beginnend HBO-bachelor Elektrotechniek demonstreert dat hij/zij in staat is om:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Programma van eisen opstellen</td>
<td>op basis van de wensen van de opdrachtgever/klant een programma van eisen (t.a.v. specificaties, certificering, prijs en levertijd) op te stellen voor een E-product/systeem/dienst dat/die voldoet aan de wensen van de opdrachtgever en realiseerbaar is</td>
</tr>
<tr>
<td>2 Keuze maken</td>
<td>beargumenteerd een keuze te maken uit een aantal bestaande of zelf bedachte concepten, oplossingen en/of verschillende implementaties</td>
</tr>
<tr>
<td>3 Projectplan opstellen</td>
<td>op basis van een programma van eisen een projectplan op te stellen</td>
</tr>
<tr>
<td>7 Voortgang bewaken</td>
<td>op basis van een projectplan de voortgang te bewaken en zonodig bij te sturen</td>
</tr>
<tr>
<td>9 Opleveren</td>
<td>aan te tonen aan de opdrachtgever of klant dat de/het geleverde product/systeem/dienst voldoet aan de specificaties</td>
</tr>
<tr>
<td>E1/5 Samenstellen</td>
<td>Het/de in het projectplan aangegeven product/systeem/dienst uit te werken tot een geheel dat samengesteld kan worden uit beschikbare deelproducten/ontwerp waarbij gebruikgemaakt wordt van beschikbare deelproducten/kennis</td>
</tr>
<tr>
<td>E2/8 Optimaliseren</td>
<td>het proces in het apparaat optimaliseren door op basis van een analyse verbeteringen aan te brengen of zonodig innovatie te starten</td>
</tr>
<tr>
<td>E3/12 Verbeteringen voorstellen</td>
<td>Op basis van een analyse van een technisch systeem verbeteringen voor te stellen</td>
</tr>
</tbody>
</table>

**Algemene competenties:**

<table>
<thead>
<tr>
<th>Kenmerk:</th>
<th>De beginnend HBO-bachelor Elektrotechniek demonstreert dat hij/zij in staat is om:</th>
</tr>
</thead>
<tbody>
<tr>
<td>17 Reflecteren</td>
<td>door reflectie op eigen gedrag, door het geven en ontvangen van terugkoppeling te formuleren wat zijn sterke/zwakke punten zijn, welke competenties hij heeft, welke rollen hij kan en wil spelen en in welke richting hij zich kan en wil ontwikkelen</td>
</tr>
<tr>
<td>19 Aanpassen</td>
<td>zich snel aan te passen aan veranderende werkomgevingen en op basis van doorzettingsvermogen prestatiegericht te werken</td>
</tr>
<tr>
<td>21 Communicatie onderhouden</td>
<td>op basis van zijn E-deskundigheid de communicatie met opdrachtgever/klant, ontwikkelaars, commerciële medewerkers en andere betrokkenen te onderhouden</td>
</tr>
<tr>
<td>22 Verantwoording afleggen</td>
<td>verantwoording af te leggen aan alle betrokkenen</td>
</tr>
<tr>
<td>E4/14 Partners kiezen</td>
<td>Leveranciers en klanten te beoordelen en kiezen</td>
</tr>
<tr>
<td>E5/20 Innoveren</td>
<td>Op basis van een goed omschreven probleem met bekende kaders te komen tot een aantal originele innovatieve concepten en/of oplossingen</td>
</tr>
</tbody>
</table>
**Definitieve opdracht omschrijving**

Het UMCG wil middels een MRI en CT scanner de cardiac output en het flow profiel meten bij verschillende hartslagen per minuut en bloeddrukken op gezonde en defecte hartkleppen. Hiervoor hebben ze een systeem nodig dat het menselijk hart (linker ventrikel) en de aorta nabootst. Via dit systeem weten ze dan hoe de situatie is en kunnen ze dit vergelijken met wat ze zien op de MRI of CT scanner.

Doel van deze opdracht is om dit systeem (verder) te ontwikkelen. De pomp moet dusdanig worden aangestuurd dat het stromingsprofiel in het menselijk hart wordt nagebootst. Tevens moeten de stromingsprofielen kunnen worden aangepast.

De bedoeling is dat de pomp ook nog voor vaatonderzoek gebruikt zal worden. Hiervoor moet het systeem dan worden aangepast.

**Plan van aanpak**

1. Verder met het concept dat in kwartaal twee is opgezet;
2. Prototype motor driver maken;
3. Formulieren inleveren;
4. Prototype mainboard (af)maken;
5. Basis voor de software voor in de mainboard maken;
6. Implementatie van WTB gedeelte en E gedeelte;
7. Naar voren gekomen problemen oplossen;
8. Eerste test in het UMCG (in MRI / CT);
9. Tussentijdse evaluatie;
10. Software verder uitdiepen en ontwikkelen;
11. Bèta PCB ontwikkelen voor mainboard;
12. Tweede test in het UMCG (in MRI?);
13. Beginnen met concept rapport;
14. Problemen die tijdens het testen naar voren zijn gekomen oplossen;
15. Eindrapport;
16. Eindevaluatie;
17. Eindpresentatie;

**Planning**

See next page.
<table>
<thead>
<tr>
<th>ID</th>
<th>Task Mode</th>
<th>Task Name</th>
<th>Duration</th>
<th>Start</th>
<th>Finish</th>
<th>Predecessors</th>
<th>February</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6-2</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>Hand in forms</td>
<td>1 day</td>
<td>Wed 8-2-12</td>
<td>Wed 8-2-12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>Graduation Project PHP</td>
<td>94 days</td>
<td>Mon 13-2-12</td>
<td>Thu 21-6-12</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>Hand in information form secr.</td>
<td>1 day</td>
<td>Thu 16-2-12</td>
<td>Thu 16-2-12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>Definitive graduation plan</td>
<td>10 days</td>
<td>Mon 20-2-12</td>
<td>Fri 2-3-12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>First meeting with teacher (company tutor)</td>
<td>5 days</td>
<td>Mon 5-3-12</td>
<td>Fri 9-3-12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>Half-way evaluation</td>
<td>5 days</td>
<td>Mon 2-4-12</td>
<td>Fri 6-4-12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>Appraisal (functioneringsgesprek)</td>
<td>5 days</td>
<td>Mon 9-4-12</td>
<td>Fri 13-4-12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>Prototype circuit for motordriver</td>
<td>15 days</td>
<td>Mon 13-2-12</td>
<td>Fri 2-3-12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>Resolve microcontroller protoboard issues</td>
<td>5 days</td>
<td>Mon 5-3-12</td>
<td>Fri 9-3-12</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>Finish microcontroller protoboard</td>
<td>5 days</td>
<td>Mon 12-3-12</td>
<td>Fri 16-3-12</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>uC software start</td>
<td>5 days</td>
<td>Mon 19-3-12</td>
<td>Fri 23-3-12</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>First E and M implementation</td>
<td>5 days</td>
<td>Mon 26-3-12</td>
<td>Fri 30-3-12</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>Resolve E and M issues</td>
<td>3 days</td>
<td>Mon 2-4-12</td>
<td>Wed 4-4-12</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>First test in MRI / CT</td>
<td>1 day</td>
<td>Thu 5-4-12</td>
<td>Thu 5-4-12</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>Advanced uC software</td>
<td>19 days</td>
<td>Fri 6-4-12</td>
<td>Wed 2-5-12</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td>Beta PCB</td>
<td>30 days</td>
<td>Fri 6-4-12</td>
<td>Thu 17-5-12</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td>Design beta PCB</td>
<td>14 days</td>
<td>Fri 6-4-12</td>
<td>Wed 25-4-12</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>Order beta PCB</td>
<td>0 days</td>
<td>Wed 25-4-12</td>
<td>Wed 25-4-12</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td>Delivery time PCB</td>
<td>14 days</td>
<td>Thu 26-4-12</td>
<td>Tue 15-5-12</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td>Solder beta PCB</td>
<td>2 days</td>
<td>Wed 16-5-12</td>
<td>Thu 17-5-12</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td>Second test in MRI?</td>
<td>2 days</td>
<td>Thu 3-5-12</td>
<td>Fri 4-5-12</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td>Resolve issues arisen during second test</td>
<td>8 days</td>
<td>Mon 7-5-12</td>
<td>Wed 16-5-12</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td>Create draft report</td>
<td>15 days</td>
<td>Tue 8-5-12</td>
<td>Mon 28-5-12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
<td>Hand in draft report</td>
<td>1 day</td>
<td>Tue 29-5-12</td>
<td>Tue 29-5-12</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
<td>Create final report</td>
<td>9 days</td>
<td>Wed 30-5-12</td>
<td>Mon 11-6-12</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
<td>Final appraisal</td>
<td>10 days</td>
<td>Mon 4-6-12</td>
<td>Fri 15-6-12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td>Hand in final report</td>
<td>1 day</td>
<td>Tue 12-6-12</td>
<td>Tue 12-6-12</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td></td>
<td>Final project presentation</td>
<td>8 days</td>
<td>Wed 20-6-12</td>
<td>Fri 29-6-12</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>