NeoMate: Movement monitoring for neonates

Yiu Man Fung (ymf112)
Benjamin Hou (bh1511)
Benjamin Moffatt (bm1810)
Vithulan Patkunan (vp1312)

1 Introduction

Patients with Cerebral palsy can suffer from muscle weakness and diminished coordination often resulting in impaired mobility. Detection of the condition is difficult at an early age due to effects only being seen when the child is struggling with initial mobility or other such progress markers. However, research suggests there is a link between the development of the neurological system that takes place during the neonatal period of an infant's life, and the incidence of the condition.

This report details the proposed solution to the project brief; 'A wearable system for monitoring movement of neonates'. The background, technical design and future developments of the created systems are discussed. In-depth program code, hardware specifications and software usage instructions are provided in the appendix.

1.1 Background

Formally classified as the first 28 days after birth, the neonatal period defines the transitional period between intrauterine and independent existence. Progress in this period is particularly significant to the development of the baby. One of the most important developments is the somatosensory system: response of the infant to stimuli found from its new environment. This system ensures information from touch, temperature and pain sensors (among others) is relayed to the motor system.

In normal development, random movements in neonates cause the somatosensory system to activate and allow the infant to react to the new stimuli appropriately (e.g. flinch from pain). As such, spontaneous movement in new-borns is essential in appropriating and fine-tuning the motor neuron system. Cerebral palsy presents itself with an ill-developed motor system. So by highlighting any differences between normal and affected infantile movement, a pattern could be discovered, leading to the possibility of early detection of Cerebral palsy. In order to do that, a system that monitors and logs spontaneous movements in neonates is required.

1.2 Project Aims

"A Wearable System for Monitoring Movement of Neonates"

It was decided the system should monitor movement of all four limbs, concentrating on flexion at the elbows and knees. The monitoring system had to be non-invasive, safe and designed in a manner that caused minimum stress in both the infants and their parents. As such, large pieces of metal were to be avoided and the familiarity of the standard baby suit made it a prime contender for being the wearable base of the system. Furthermore, the whole kit had to be small and lightweight so as not to restrict movement of the baby. Finally it had to be easy to clean in order to use again on different babies without the risk of spreading infection.

A secondary aim was to exclude ferrous materials and hence make it MRI compatible.

1.3 Initial Ideas

Input Process: A monitoring system was designed using fluid-filled tubes attached along the limbs of a baby suit. Bending of a limb was to cause the tube to contract, causing compression within, and the fluid volume to change. This change in volume would be recorded by measuring levels in a reservoir connected, but kept apart, from the suit. This design avoided the use of metals so was MRI compatible. However, it was found that bending at such a small scale didn't produce a large enough change in fluid levels to be measured consistently.

Output Process: An animated on-screen model of a baby was to be used to track the output of an electronic monitoring system, consisting of flex sensors attached to a baby. When the real-world infant would move a limb, the corresponding limb of the computerised model would move to the same degree of flexion. However, the movement data would have needed filtering and a higher sampling rate in order to move the model smoothly. Furthermore, within the scope of the project, the mapping of data to realistic human-like movement was unfeasible. After further iterations and a re-organisation of project aim priorities, NeoMate was born.

2 Solution: NeoMate

NeoMate monitors limb movement in neonates with accompanying tailor-made software that carries out data logging. The movement data for each limb is stored for later analysis by both medical professionals and parents/carers at home.

Two products have been developed for the NeoMate brand; NeoMate: Home and NeoMate: Hospital. Both systems are based around a suit with flex sensors embedded; the home version's electronics have been designed to interface with hardware and software widely used by the home user, whilst the hospital version contains more complex networking and data analysis features.

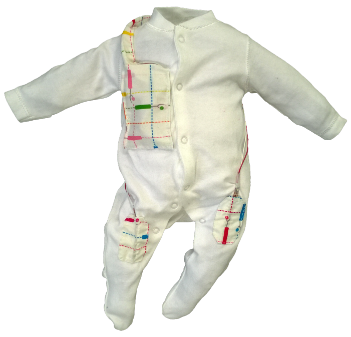


Figure 1: NeoMate suit with electronics installed

2.1 NeoMate: Home

Designed for monitoring of babies at home, this is a low cost baby monitoring suit with four flex sensors – one on each of the arms and legs. The suit can either be powered by rechargeable Li-On battery or USB and can communicate via Bluetooth or Serial over USB. The suit is designed to work with the NeoMate: Home monitoring and logging software that runs in Microsoft Office Excel, a standard installation on most consumer computers. The software can log movements for up to 24 hours, with live graphs and auto-updating diagrams to display limb movement.

2.2 NeoMate: Hospital

Designed for monitoring many babies simultaneously in a hospital environment, this suit uses the same four flex sensors as the home version, yet features an XBee wireless device and mBed platform integrating an ARM Cortex-M3 processor. Four flex sensors are attached to the baby suit which can be powered by Li-On battery or USB. Data from each suit is directly uploaded to the Cloud. Multiple practitioners can access data and history of all NeoMate patients. Data can be recorded and stored for weeks at a time.

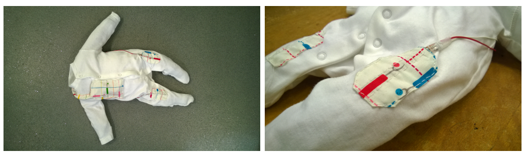
3 Technical Design

The working principle of NeoMate is that when infants wear the baby suit and bend their limbs, the embedded flex sensors will be bent accordingly, which causes a change in resistance of the sensor. After considering different design parameters, four major design requirements were decided for NeoMate: the baby suit has to be able to measure limb movements, be safe, aesthetically appealing and easy to clean.

3.1 Suit Design

The flex sensors on the baby suit are placed in specific locations and orientations such that any flex in the sensors will only be caused by the interested limb movements. Moreover, the wires connecting the flex sensors to the processing circuit board are fixed in position, so that they cannot be displaced and strangle the baby.

Baby suits often get dirty; therefore NeoMate is designed to be machine washable, with detachable flex sensors and electronic components. This feature is achieved by sewing pockets and channels for the flex sensors, rather than sewing the sensors directly onto the baby suit. These pockets at each joint fit the sensors snugly to ensure they do not fall out of position. All of the channels are covered with patches made with patterned fabrics ensuring the baby suit will look appealing to parents and hospitals purchasing NeoMate.


(a) NeoMate: Home suit with wide joint patches

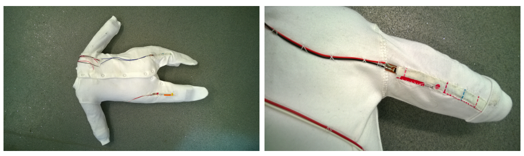
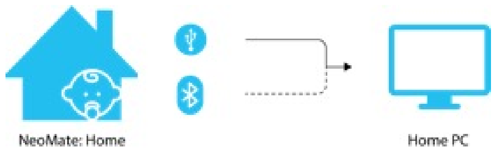

(b) NeoMate: Hospital suit with slim joint patches

Figure 2: NeoMate Home and Hospital Baby Suits

3.2 NeoMate: Home

3.2.1 Hardware

The baby suit has a front pocket for storage of the circuit board. The board has been prototyped on strip board and consists of an Arduino Micro, soldered flex sensor connections, 22 KΩ resistors to create a voltage divider for each sensor and female header pins for attachment of a HC-06 Bluetooth module. In the prototype, the Arduino Micro is powered by micro-USB, which also carries the data via serial to the host computer. It is also possible to power the device with a Lithium-Ion rechargeable battery and send data over Bluetooth serial to the computer. The Bluetooth HC-06 module can just be plugged into the provided header pins and paired with the computer. The correct COM port for Bluetooth must subsequently be selected in the provided Excel program.



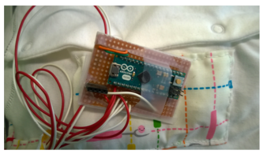


Figure 3: (Left) NeoMate: Home communications; (Right) Prototype board with Arduino Micro, hardwired sensors and female header pins to attach HC06 Bluetooth module

3.2.2 Firmware

The Arduino Micro uses the Arduino boot-loader. Therefore a sketch (microcontroller program) for NeoMate: Home has been developed using the Arduino Integrated Development Environment (IDE). The sketch sets up serial communication for communicating via USB connection and establishes a software serial protocol for sending commands to the Bluetooth module. Within the main loop, an 'analog read' of the on-board 10-bit ADC reads the four sensor voltage divider inputs. The analogue readings are scaled individually for each sensor to provide an output between 0-100. The scaled readings are sent via serial in CSV format at a BAUD rate of 9600. The full sketch is provided in the appendix.

3.2.3 Software

A data logging and live monitoring program has been developed using Visual Basic for Applications embedded within Microsoft Office Excel. The program takes serial data from a COM port provided by the user and inserts the data into a recording sheet. This sheet retains data for up to 24 hours. The software queries this input data to overlay the live sensor values onto an outline of a body within an overlaid window. Squares at each limb joint change colour from yellow->green->orange->red depending on how much the limb is bent. A separate sheet queries the stored data for the last 15 minutes, hour and all time to provide the average movement and live graphs displaying historical limb movement. The serial communication is based on a Visual Basic script produced by PLX-DAQ. For screenshots of the created software, see appendix.

3.3 NeoMate: Hospital

3.3.1 System Overview

As the NeoMate: Hospital model will be used to monitor multiple neonates wirelessly, the system would need to be in a star network configuration. This involves multiple end devices with a single co-ordinator. The end device will perform data acquisition and send it to the co-ordinator, where it can perform data manipulation and forward the values to a logging system.

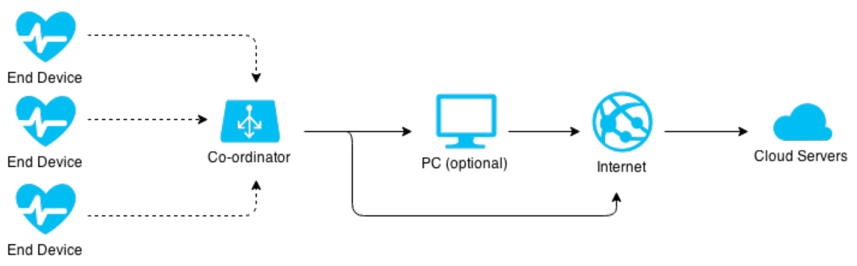


Figure 4: Neomate: Hospital communications

3.3.2 Hardware

The wireless hardware of choice is the XBee by DIGI (Part: XBEE-PRO-S2B) as it utilises a Zigbee chip for communication. Zigbee is particularly ideal for this purpose as it is designed for low power consumption whilst maintaining high data throughput.

The XBee can be configured using DIGI's X-CTU software, which can be used to upload various DIGI's in-house firmware onto the XBee chip. Different firmware gives the XBee different functionality (e.g. endpoint, router or co-ordinator). After uploading the firmware, the XBee can be configured via AT commands through the Rx-Tx pins.

The XBee board has a Freescale MC9S08QE32CFT microprocessor on board along with a Silicon Lab EM250 Zigbee radio chip. It is possible to develop firmware for the microprocessor directly, using Freescale CodeWarrior IDE and DIGI's USB Multilink Debugger. This will unlock lower level functionality and IO, but with the given timescale, this was not possible.

The companion microcontroller to go with the XBee radios is the mbed microcontroller, which has a 32bit ARM Cortex-M3 processor (Part: LPC1768). The mbed is capable of utilising many serial ports as well as Ethernet and USB communication. Combined with the mbed application board, which contains a multitude of external accessories such as an LCD panel, XBee socket, Ethernet socket, sensors and more, this platform is ideal to power the hub. Firmware for the mbed is developed in C/C++ on ARM's web-based IDE, which is then compiled by ARM servers to produce a .bin file. The .bin file is then uploaded onto the mbed to run.

3.3.3 End Device

The End Device is built around one XBee module. From the schematic, pins 17-20 of the XBee can be utilised as ADCs with a max sampling rate of 1KHz. The XBee is therefore flashed with a "ROUTER AT" firmware (router firmware and end device firmware are identical apart from router firmware are not allowed to sleep and they are able to relay data to the co-ordinator via other routers) and have all 4 pins enabled as ADCs with a sampling rate of 5Hz. 5 samples a second was chosen to be sufficient resolution without congesting the network traffic. The readings of all 4 ADCs are sent in one packet to the coordinator.

3.3.4 Co-ordinator

The XBee on the co-ordinator module is flashed with "COORDINATOR API" firmware. This XBee resides as one of the components on the mbed application board and communicates via serial Rx-Tx.

The firmware is developed in a way so that it would read and dissect the incoming data packet from the end devices and perform various data manipulation functions. It is possible to implement a digital filter to filter out high frequencies, but with a low sampling rate of 5Hz, this was not needed. The values from the incoming ADCs are printed on the LCD for debugging as well as sent via USB serial (in CSV format) so that it can be logged/-graphed by a computer. Optionally the mbed application board can also be connected directly to internet via Ethernet. Therefore data can be sent to cloud servers directly via HTTP POST/GET methods.

3.3.5 PC (Optional)

If the co-ordinator is connected to a PC, the data can be logged by the same software as for NeoMate: Home. The incoming data can also be forwarded to cloud servers by a simple python script. The python script can be found in the appendix.

3.3.6 Cloud Services

AWS is used to host an Ubuntu machine with thingspeak installed. thingspeak is an open-source real-time data collection platform and can be used to instantiate multiple monitors (one monitor per wireless sensor) where each monitor is able to accommodate multiple feeds.

4 Future Developments

Given the short timescale of the project, several features were not implemented for NeoMate and could be explored and adopted as future improvements.

First and foremost, more flex sensors and an accelerometer could be placed on the baby suit to measure other baby movements such as breathing or body orientation. This would allow more data to be collected for diagnosis or research.

Secondly, the XBee unit in NeoMate: Hosptial and the strip board created for NeoMate: Home should be replaced by a printed circuit board, which has a smaller size and weight and could more easily incorporated into the baby suit.

Thirdly, in addition to the bending angle, flex sensor readings also provide information on the frequency of limb movements, which could be analysed and used to alert of potential seizure.

Lastly, the system of NeoMate has the potential to be modified and applied to other patients, apart from neonates, who also need movement monitoring.

5 Conclusion

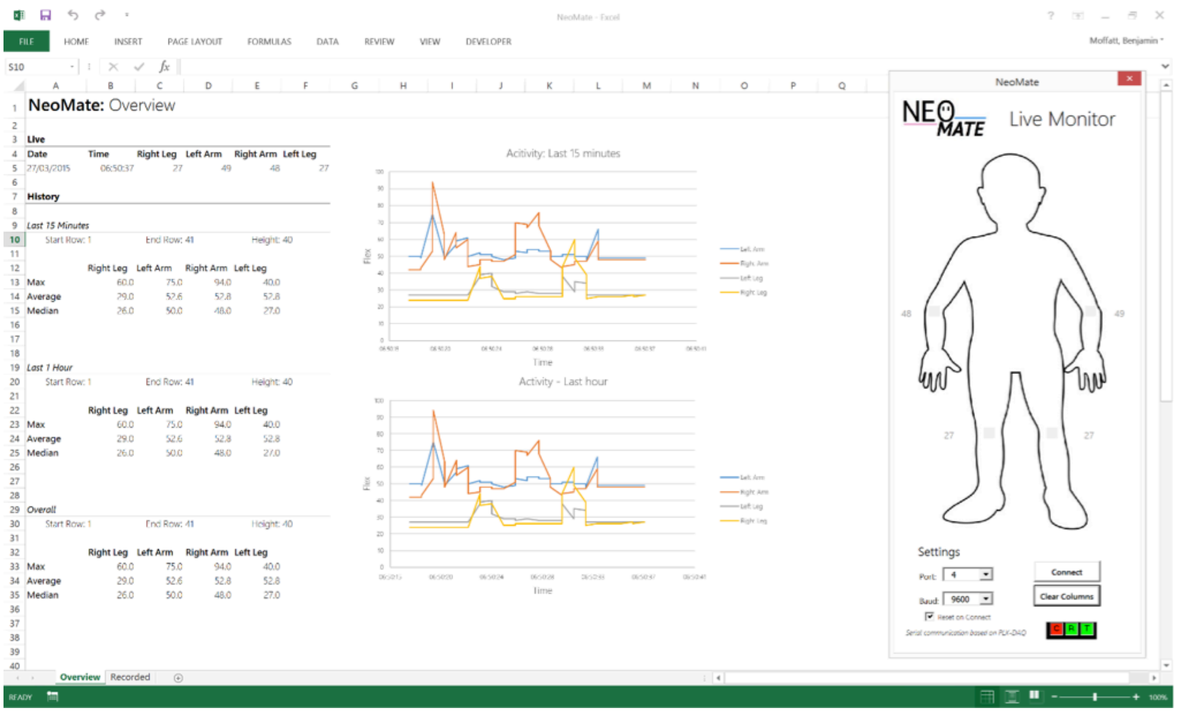
This project has successfully delivered two working systems for monitoring the movement of neonates. Using flex sensors attached to a baby suit, live monitoring of babies can be affordably achieved at home with recording for up to 24 hours. For hospitals, a network of NeoMates can be used to monitor multiple babies, with functionality to upload the data to cloud servers. The project has also demonstrated potential for progression into other forms of patient monitoring such as breathing and seizure alert.

6 Appendix

6.1 NeoMate: Home

6.1.1 Excel and Visual Basic program

The following screenshots demonstrate the NeoMate: Home software. The program has been developed in a Microsoft Excel macro enabled spreadsheet with an interactive 'Live Monitor' overlaid form.



NeoMate: Overview

Live

Date	Time	Right Leg	Left Arm	Right Arm	Left Leg
27/03/2015	06:50:37	27	49	48	27

History

Last 15 Minutes

Start Row: 1 End Row: 41 Height: 40

	Right Leg	Left Arm	Right Arm	Left Leg
Max	60.0	75.0	94.0	40.0
Average	29.0	52.6	52.8	52.8
Median	26.0	50.0	48.0	27.0

Last 1 Hour

Start Row: 1 End Row: 41 Height: 40

	Right Leg	Left Arm	Right Arm	Left Leg
Max	60.0	75.0	94.0	40.0
Average	29.0	52.6	52.8	52.8
Median	26.0	50.0	48.0	27.0

Overall

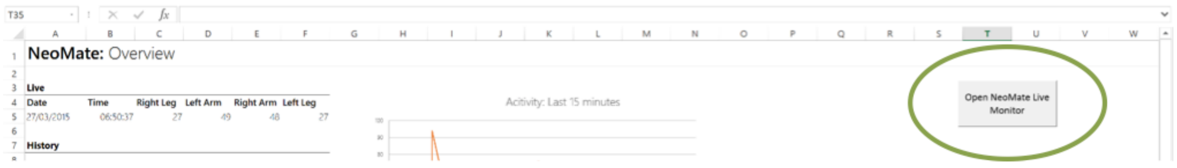
Start Row: 1 End Row: 41 Height: 40

	Right Leg	Left Arm	Right Arm	Left Leg
Max	60.0	75.0	94.0	40.0
Average	29.0	52.6	52.8	52.8
Median	26.0	50.0	48.0	27.0

Figure 5: NeoMate: Home overview screen (right) with numerically processed values over the last 15 minutes, hour and 24 hours, live updating graphs (centre) and an overlaid window 'NeoMate Live Monitor'

The Arduino Micro communicates with the spreadsheet via serial communication, received from the user selected COM port in the 'Live Monitor' form. This port can either be the computer's Bluetooth or USB port. The serial communication uses drivers produced by Parallax in their PLX DAQ software, therefore these must be installed before use.

The 'Live monitor' form shows flex in two ways; (1) with numerical values next to each joint and (2) overlaid squares on each joint that change colour from green to yellow to red depending on how much each limb is bent. The 'Live Monitor form opens on startup. If it is is closed by the user, it can be reopened with a button behind the window on the sheet:



NeoMate: Overview

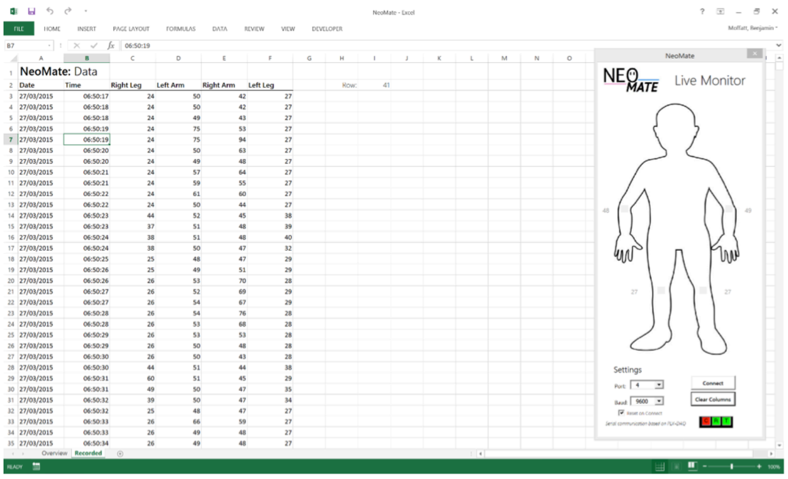
Live

Date	Time	Right Leg	Left Arm	Right Arm	Left Leg
27/03/2015	06:50:37	27	49	48	27

History

Figure 6: Button location

Figure 7 shows the second sheet that stores the sensor data received from the Arduino Micro, updated twice a second. The data is stored sequentially, therefore the last recorded value is at the bottom. The graphs in the overview sheet dynamically update for every update of the data in this sheet.



NeoMate: Data

Date	Time	Right Leg	Left Arm	Right Arm	Left Leg
27/03/2015	06:50:17	24	50	42	27
27/03/2015	06:50:18	24	50	42	27
27/03/2015	06:50:18	24	49	43	27
27/03/2015	06:50:19	24	75	53	27
27/03/2015	06:50:19	24	75	94	27
27/03/2015	06:50:20	24	50	63	27
27/03/2015	06:50:20	24	49	48	27
27/03/2015	06:50:21	24	57	64	27
27/03/2015	06:50:21	24	59	55	27
27/03/2015	06:50:22	24	61	60	27
27/03/2015	06:50:22	24	50	44	27
27/03/2015	06:50:23	44	52	45	38
27/03/2015	06:50:23	37	51	48	39
27/03/2015	06:50:24	38	51	48	40
27/03/2015	06:50:24	38	50	47	32
27/03/2015	06:50:25	25	48	47	29
27/03/2015	06:50:26	25	49	51	29
27/03/2015	06:50:26	26	53	70	28
27/03/2015	06:50:27	26	52	69	29
27/03/2015	06:50:27	26	54	67	29
27/03/2015	06:50:28	26	54	76	28
27/03/2015	06:50:28	26	53	68	28
27/03/2015	06:50:29	26	53	53	28
27/03/2015	06:50:29	26	50	48	28
27/03/2015	06:50:30	26	50	43	28
27/03/2015	06:50:30	44	51	44	38
27/03/2015	06:50:31	60	51	45	29
27/03/2015	06:50:31	49	50	47	35
27/03/2015	06:50:32	39	50	47	34
27/03/2015	06:50:32	25	48	47	27
27/03/2015	06:50:33	26	66	59	27
27/03/2015	06:50:33	26	49	48	27
27/03/2015	06:50:34	26	49	48	27

How: 41

NEO MATE Live Monitor

Settings

Port: 4

Baud: 9600

Reset on Connect

Serial communication based on PLX-DAQ

Figure 7: NeoMate: Home recorded sheet storing the historical data for each joint. This data is dynamically queried to update the live graphs; showing an output for the previous 15 minutes and hour.

6.1.2 Baby suit electronics

The circuit has been prototyped on strip board for use in the prototype suit (see figure 3). The prototype board has the sensors directly soldered to the board with header pins to attach the Bluetooth module. It has been padded to ensure there are no sharp parts that could harm injure the baby. For commercial use, rather than strip board, a 43x77mm printed circuit board (PCB) has been designed:

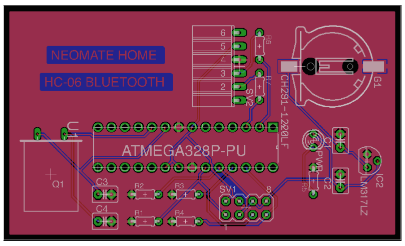


Figure 8: Printed circuit board design for NeoMate: Home

The schematic for the above design:

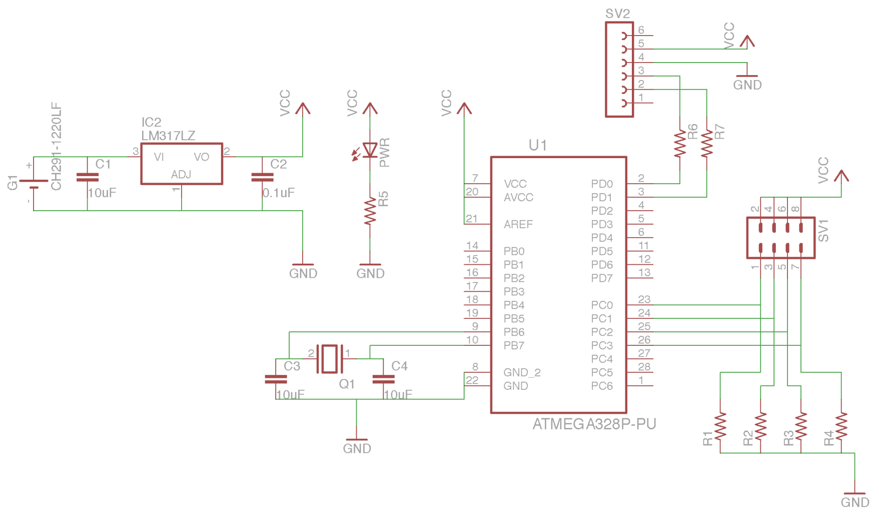


Figure 9: Circuit diagram for NeoMate: Home PCB

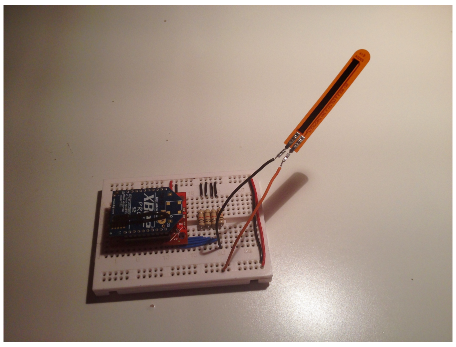
6.1.3 Arduino Code

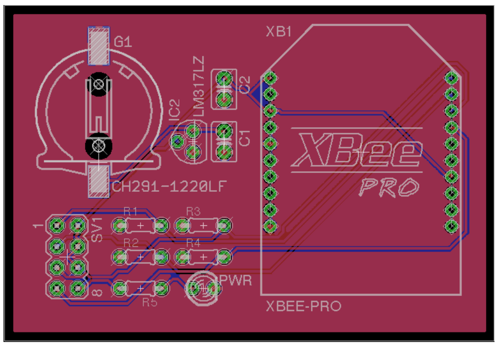
```
#include <SoftwareSerial.h>

// Define which analogue pins the flex sensors are attached to
int flexSensorPin1 = A0; //analog pin 0
int flexSensorPin2 = A1; //analog pin 1
int flexSensorPin3 = A2; //analog pin 2
int flexSensorPin4 = A3; //analog pin 3
// Setup a software serial output on pins 10 and 11
SoftwareSerial mySerial(10, 11); // RX, TX

void setup(void) {
  Serial.begin(9600); // set up for output to NeoMate spreadsheet
  mySerial.begin(9600); // Bluetooth output software serial start
}

void loop() {

  int flexSensorReading1 = analogRead(flexSensorPin1); // setup integer to store flex
      sensor output
  int flexSensorReading2 = analogRead(flexSensorPin2);
  int flexSensorReading3 = analogRead(flexSensorPin3);
  int flexSensorReading4 = analogRead(flexSensorPin4);

  /*  Flex sensors output is theoretically between 0 and 1024 due to the ADC on the
      Arduino Micro being 10bit (2^10=1024). Each flex sensor does not use this whole
      range: the first flex sensor has a range between 170 and 350. This value is mapped
      between 0 and 100: */
  flexSensorReading1 = map(flexSensorReading1,170, 350, 100, 0);
  flexSensorReading2 = map(flexSensorReading2,110, 310, 100, 0);
  flexSensorReading3 = map(flexSensorReading3,130, 350, 100, 0);
  flexSensorReading4 = map(flexSensorReading4,170, 360, 100, 0);

  /*  The sensor readings are output as a line in CSV format, headed by a DATA command
      (required for VBA code) DATE and TIME are replaced in Excel dynamically.       */

  // Wired serial output
  Serial.print("DATA,DATE,TIME,");
  Serial.print(flexSensorReading1);
  Serial.print(",");
  Serial.print(flexSensorReading2);
  Serial.print(",");
  Serial.print(flexSensorReading3);
  Serial.print(",");
  Serial.println(flexSensorReading4);
  // Bluetooth serial output
  mySerial.print("DATA,DATE,TIME,");
  mySerial.print(flexSensorReading1);
  mySerial.print(",");
  mySerial.print(flexSensorReading2);
  mySerial.print(",");
  mySerial.print(flexSensorReading3);
  mySerial.print(",");
  mySerial.println(flexSensorReading4);

  delay(500); //just here to slow down the output for two samples a second
}
```

6.2 NeoMate: Hospital

6.2.1 NeoMate: Hospital Remote Sensor

The NeoMate Hospital Remote sensor will be attached to the patient, its function is to gather recordings from the ADC and transmit the values to the coordinator via the xbee.


(a) Breadboard prototype



(b) PCB Layout

Figure 10: NeoMate Hospital Remote Sensor

Figure 10a is the breadboard prototype, it is powered via USB using SparkFun's XBee Explorer Regulated board (Part: WRL-11373). The board is setup to accommodate 4 sensors (using a potential divider circuit), only one sensor is connected in the picture.

Figure10b is the breadboard circuit redesigned in CadSoft EAGLE PCB Design software. The PCB is 60x40mm in size and can be further minimised by utilising SMD components and multilayering. The PCB is powered by a single 3.3v coin cell battery (top-left component) and has 8 header pins (bottom-left) to connect the flex sensors.

The schematic of the board can be seen in Figure11 below.

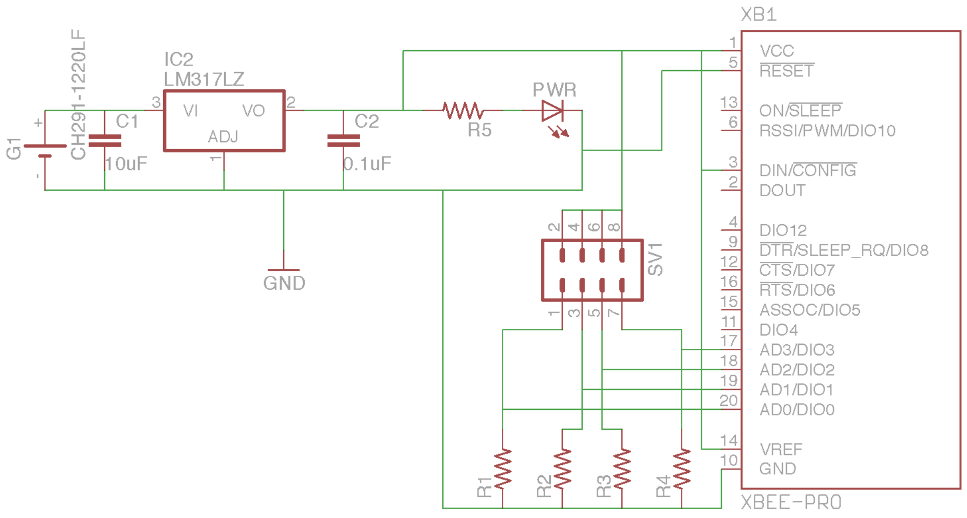
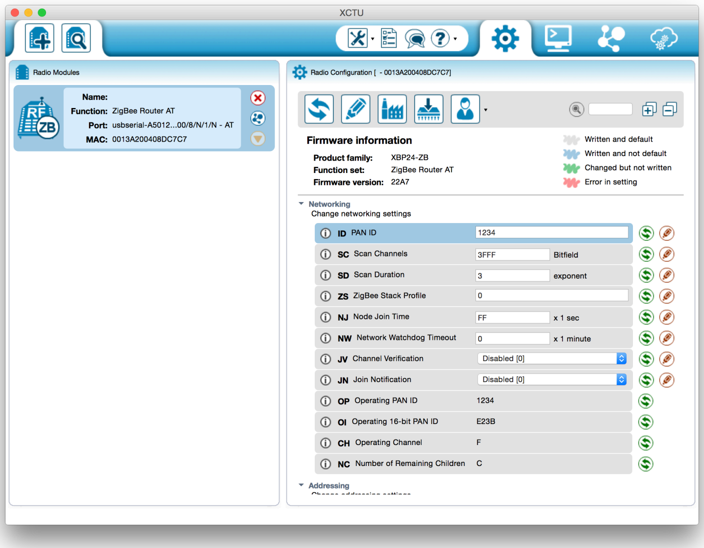


Figure 11: NeoMate: Hospital Schematic

6.2.2 DIGI's XCTU

The XBee is configured by DIGI's XCTU program, Figure12 shows the screenshot of the program. This utility can be used to flash the XBee with DIGI's in-house firmware, configure the XBee by assigning AT settings as well as perform network diagnostic such as packet analysis.

The Router XBee and Coordinator XBee used in this project are configured with AT settings listed below in XML format. The full AT command list can be found on DIGI's website (see References)



Name:
Function: ZigBee Router AT
Port: usbserial-A5012...00/8/N/1/N - AT
MAC: 0013A200408DC7C7

Firmware information

Product family:	XBP24-ZB
Function set:	ZigBee Router AT
Firmware version:	22A7

Written and default
Written and not default
Changed but not written
Error in setting

Networking
Change networking settings

ID	PAN ID	1234	
SC	Scan Channels	3FFF	Bitfield
SD	Scan Duration	3	exponent
ZS	ZigBee Stack Profile	0	
NJ	Node Join Time	FF	x 1 sec
NW	Network Watchdog Timeout	0	x 1 minute
JV	Channel Verification	Disabled [0]	
JN	Join Notification	Disabled [0]	
OP	Operating PAN ID	1234	
OI	Operating 16-bit PAN ID	E23B	
CH	Operating Channel	F	
NC	Number of Remaining Children	C	

Addressing

Figure 12: NeoMate: Hospital Schematic

XBee Router Settings

```
<?xml version="1.0" encoding="UTF-8"?>

<data>
  <profile>
    <description_file>XBP24-ZB_22A7.xml</description_file>
    <settings>
      <setting command="ID">1234</setting>
      <setting command="SC">3FFF</setting>
      <setting command="SD">3</setting>
      <setting command="ZS">0</setting>
      <setting command="NJ">FF</setting>
      <setting command="NW">0</setting>
      <setting command="JV">0</setting>
      <setting command="JN">0</setting>
      <setting command="DH">0</setting>
      <setting command="DL">0</setting>
      <setting command="NI">0x20</setting>
      <setting command="NH">30</setting>
      <setting command="BH">0</setting>
      <setting command="AR">FF</setting>
      <setting command="DD">30000</setting>
      <setting command="NT">3C</setting>
      <setting command="NO">3</setting>
      <setting command="CR">3</setting>
      <setting command="SE">E8</setting>
      <setting command="DE">E8</setting>
      <setting command="CI">11</setting>
      <setting command="PM">1</setting>
      <setting command="EE">0</setting>
      <setting command="EO">0</setting>
      <setting command="KY"></setting>
      <setting command="BD">3</setting>
      <setting command="NB">0</setting>
      <setting command="SB">0</setting>
      <setting command="RO">3</setting>
      <setting command="D7">1</setting>
      <setting command="D6">0</setting>
      <setting command="CT">64</setting>
      <setting command="GT">3E8</setting>
      <setting command="CC">2B</setting>
      <setting command="SM">0</setting>
      <setting command="SN">1</setting>
      <setting command="SO">0</setting>
      <setting command="SP">20</setting>
      <setting command="ST">1388</setting>
      <setting command="PO">0</setting>
      <setting command="D0">2</setting>
      <setting command="D1">2</setting>
      <setting command="D2">2</setting>
      <setting command="D3">2</setting>
      <setting command="D4">0</setting>
      <setting command="D5">1</setting>
      <setting command="P0">1</setting>
      <setting command="P1">0</setting>
      <setting command="P2">0</setting>
      <setting command="PR">1FFF</setting>
      <setting command="LT">0</setting>
      <setting command="RP">28</setting>
      <setting command="DO">1</setting>
      <setting command="IR">C8</setting>
      <setting command="IC">0</setting>
      <setting command="V+">0</setting>
    </settings>
  </profile>
</data>
```

XBee Coordinator Settings

```
<?xml version="1.0" encoding="UTF-8"?>

<data>
  <profile>
    <description_file>XBP24-ZB_21A7.xml</description_file>
    <settings>
      <setting command="ID">1234</setting>
      <setting command="SC">3FFF</setting>
      <setting command="SD">3</setting>
      <setting command="ZS">0</setting>
      <setting command="NJ">FF</setting>
      <setting command="DH">0</setting>
      <setting command="DL">FFFF</setting>
      <setting command="NI">0x20</setting>
      <setting command="NH">1E</setting>
      <setting command="BH">0</setting>
      <setting command="AR">FF</setting>
      <setting command="DD">30000</setting>
      <setting command="NT">3C</setting>
      <setting command="NO">3</setting>
      <setting command="CR">3</setting>
      <setting command="PM">1</setting>
      <setting command="EE">0</setting>
      <setting command="EO">0</setting>
      <setting command="KY"></setting>
      <setting command="NK"></setting>
      <setting command="BD">3</setting>
      <setting command="NB">0</setting>
      <setting command="SB">0</setting>
      <setting command="D7">1</setting>
      <setting command="D6">0</setting>
      <setting command="AP">1</setting>
      <setting command="AO">0</setting>
      <setting command="SP">20</setting>
      <setting command="SN">1</setting>
      <setting command="D0">1</setting>
      <setting command="D1">0</setting>
      <setting command="D2">0</setting>
      <setting command="D3">0</setting>
      <setting command="D4">0</setting>
      <setting command="D5">1</setting>
      <setting command="P0">1</setting>
      <setting command="P1">0</setting>
      <setting command="P2">0</setting>
      <setting command="PR">1FFF</setting>
      <setting command="LT">0</setting>
      <setting command="RP">28</setting>
      <setting command="DO">1</setting>
      <setting command="IR">0</setting>
      <setting command="IC">0</setting>
      <setting command="V+">0</setting>
    </settings>
  </profile>
</data>
```

6.2.3 NeoMate: Hospital Co-ordinator

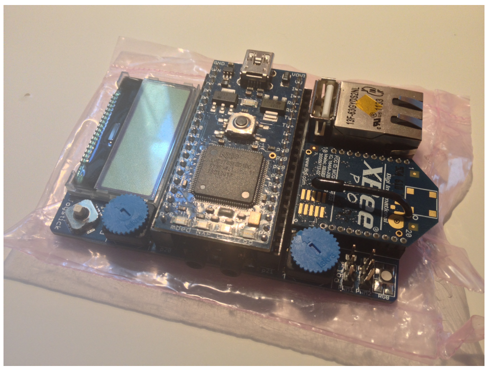


Figure 13: NeoMate: Hospital Coordinator board

Figure 13 shows the coordinator board of NeoMate Hospital. Data from the remote XBees are received here. The mbed reads the data from the RxTx ports and processes it appropriately. The code below is uploaded onto the mbed.

The code is sectioned using preprocessor directives. This way, certain features can be enabled or disabled without having to comment out sections of code.

```
/* Feature Switches */
#define bEnableCalibration 0
#define bEnableEthernet 0
#define bEnableBluetooth 0
#define bEnableLCD 1

/* includes */
#include "mbed.h"
#include "XBee.h"

Serial pc(USBTX, USBRX); //Initalise PC serial comms
Serial xbee(p9,p10);     //Initalise xbee_lib varName(rx,tx)
DigitalOut xbee_rst(p30);

#if bEnableLCD
#include "C12832_lcd.h"  // Include for LCD code
C12832_LCD lcd;          // Initialize LCD Screen
#endif

#if bEnableBluetooth
Serial BT(p17,p18);      //Initalise Bluetooth
#endif

#if bEnableEthernet
#include "EthernetInterface.h"
#include "HTTPClient.h"
char* thingSpeakUrl = "http://52.16.59.200:3000/update";
char* thingSpeakKey = "F8OGU7BBSJKKFA3E";
char urlBuffer[256];
HTTPClient http;
char str[512];
#endif

int intMap(int x, int inMin, int inMax, int outMin, int outMax)
{
    return (x-inMin)*(outMax-outMin)/(inMax-inMin)+outMin;
}

int main()
{
#if bEnableEthernet
    EthernetInterface eth;
    eth.init(); //Use DHCP
    eth.connect();
    printf("IP Address is %s\n", eth.getIPAddress());
#endif

    xbee_rst = 0;
    wait_ms(1000);
    xbee_rst = 1;
    wait_ms(1000);

#if bEnableLCD
    lcd.printf(" ...XBEE READY... \n");
#endif
```

```
    // Example Packet
    // 7E 00 18 92 00 13 A2 00 40 8D C7 C7 4F 88 01 01 00 00 0F 02 0C 02 1F 02 14 00 76 BA

    while(1) {

        uint8_t buff[26];

        if (xbee.readable() && xbee.getc() == 0x7e) {
            uint8_t frameSize_H = xbee.getc();
            uint8_t frameSize_L = xbee.getc();
            uint16_t frameSize = frameSize_L + (frameSize_H << 8);
            xbee.gets((char*)buff,frameSize+2);

            /*
            pc.printf("%x, \t %x, \t %x, \t",0x7e,frameSize_H,frameSize_L);
            for (int i=0; i<frameSize+2; i++) {
                pc.printf("%x, \t",buff[i]);
            }
            pc.printf("\n");
            */

            //N.B. buff[4] to buff[13] contains the remote xbee Serial ID
            //Since there is only one other remote xbee in the network, ID checking is
                omitted

            uint8_t ADC0_H = buff[16];
            uint8_t ADC0_L = buff[17];
            uint8_t ADC1_H = buff[18];
            uint8_t ADC1_L = buff[19];
            uint8_t ADC2_H = buff[20];
            uint8_t ADC2_L = buff[21];
            uint8_t ADC3_H = buff[22];
            uint8_t ADC3_L = buff[23];

            uint16_t ADC0 = ADC0_L + (ADC0_H << 8);
            uint16_t ADC1 = ADC1_L + (ADC1_H << 8);
            uint16_t ADC2 = ADC2_L + (ADC2_H << 8);
            uint16_t ADC3 = ADC3_L + (ADC3_H << 8);

#if bEnableCalibration
            ADC0 = intMap(ADC0, 100, 220, 1, 512); //H: 219 L: 100
            ADC1 = intMap(ADC1, 10, 400, 1, 512); //H: 360 L: 10
            ADC2 = intMap(ADC2, 40, 120, 1, 512); //H: 107 L: 40
            ADC3 = intMap(ADC3, 10, 70, 1, 512); //H: 64  L: 10
#endif

            pc.printf("DATA,%d,%d,%d,%d\n", ADC0, ADC1, ADC2, ADC3);

#if bEnableLCD
            lcd.cls();
            lcd.locate(0,0);
            lcd.printf("ADC0 \t ADC1 \t ADC2 \t ADC3\n");
            lcd.printf("%d \t %d  \t %d  \t %d", ADC0, ADC1, ADC2, ADC3);
#endif

#if bEnableBluetooth
            BT.printf(",%d,%d,%d,%d,\n", ADC0, ADC1, ADC2, ADC3);
            BT.printf("%d,%d,%d,%d\n", ADC0, ADC1, ADC2, ADC3);
#endif

#if bEnableEthernet
            urlBuffer[0] = 0;
            sprintf(urlBuffer, "%s?key=%s&field1=%d&field2=%d&field3=%d&field4=%d",
                thingSpeakUrl, thingSpeakKey, ADC0, ADC1, ADC2, ADC3);
            printf("%s\n",urlBuffer);
            http.get(urlBuffer, str,128);
#endif

        }
    }
}
```

6.2.4 PC and Cloud Services

The Data from the coordinator can be sent directly via the mbed ethernet port, it can also be sent via a PC (if the PC is connected to the internet). Using a simple Python script (below), the incoming data from the coordinator (via USB) is read and then sent to the cloud.

```
#For MacOS and Linux
import serial
import urllib2

ser = serial.Serial('/dev/tty.usbserial', 9600)

server = str('52.17.32.235:3000')
key = str('F8OGU7BBSJKKFA3E')

while True:
    message = str(ser.readline())
    #print(message)
    b = message.split(",")
    c = "http://%s/update?key=%s&field1=%s&field2=%s&field3=%s&field4=%s" %
        (server,key,b[0],b[1],b[2],b[3])
    response = urllib2.urlopen(c)
    print response.read()
```

Upon receiving, the server will log the data for all time and present the graph in a web GUI.

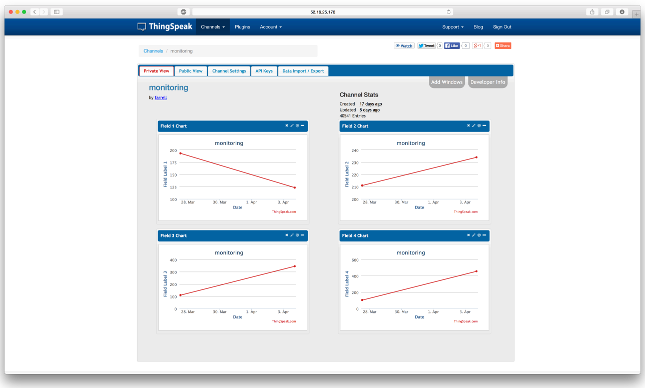


Figure 14: Cloud Server Logging

7 References

A xbee manual
http://ftp1.digi.com/support/documentation/90001192_a.pdf

B Schematic
http://ftp1.digi.com/support/documentation/3001115101_b.pdf

C xbee ADC sample rate
http://www.digi.com/support/kbase/kbaseresultdetl?id=2180

D xbee quick reference guide
http://www.tunnelsup.com/images/XBee-Quick-Reference-Guide.pdf

E xbee AT Command List
http://examples.digi.com/wp-content/uploads/2012/07/XBee_ZB_ZigBee_AT_Commands.pdf

F Parallax PLX DAQ
https://www.parallax.com/downloads/plx-daq