****

**Module Title: Embedded Systems**

**Module Code: CSE6012**

**Module Level: 6**

**Module Leader: Pr. Hasan Al-Saedy**

**Student Name: Mohammad Bappy Chowdhary**

**Student ID: 45765**

**EMBEDDED SYSTEM ENVIRONMENT:**

Embedded system, an architecturally design to run or minimize the work effort of a system which can be a microcontroller or can be a microchip and they are programmable to manage the workflow of the system.

Comparing with a personal computer a specific embedded device will be limited to its software and hardware functionality. Reliability requirements indeed a major fact for any embedded system such as a car engine controller malfunction can cause a huge damage.

A typical embedded system contains a microprocessor, technical way we call it digital signal processor, or microcontrollers. As these are designed to perform specific task, they are optimised by size and therefore production cost are less, which are quiet economical.

Traffic signals, medical devices, portable devices and robotic designs are typically performed with the help of embedded systems.

**RASPBERRY PI:**

A tiny computer system called Raspberry Pi introduce to students to gain the knowledge of basic programming skills. But the purpose of the device went beyond of thinking regarding its portability and performance on embedded system applications.

Configuring the Raspberry Pi (RPi), this tiny computer runs on a 700 MHz ARM processor and allows over clock up to 1000MHz which doesn’t void the warranty. It has two models, model A and model B. The main differences between model A and B are A has 256 Mb of Ram and no physical Ethernet port and a single USB port, where B has 512 Mb of ram, an Ethernet port and dual USB port. In this project we are going to use the model B.

Architecturally the Raspberry Pi includes a HDMI connection, a slot for camera and 26 pins (Model B) GPIO (General Purpose Input Output) which allows connecting various external embedded devices such as motor drivers or stepper drivers with 3v and 5v power supplies from the GPIO pins.

**HARDWARE REQUIREMENTS:**

In this project we are going to build a heart rate sensor environment to measure the heart rate as BPM (Beat per Minute). Using Raspberry pi as the system (heart rate monitor) controller with the help of python GPIO library coding such as calculating IBI (Inter beat Interval) and the mini pulse sensor amped and a microchip named MCP3008 to convert the analog pulse sense from the sensor into digital format to view in the screen, we can build our mini ECG system.

**1. Pulse Sensor Amped:**

A small heart shaped heart rate sensor. Front part of the sensor with white logo, contact with skin. In the middle of the sensor a round small hole which include an LED from the back and there is a square underneath it which is an ambient light sensor which we normally find in cell phones to adjust the brightness in various light conditions.

The LED shines light into the capillary tissue (fingertips or earlobes) while sensor reads the light and bounce back. We can fix the rate by counting it in 2 milliseconds.

It comes with colour coded 3 male header colour coded 24” cable which will very useful to use with help of a breadboard where soldering is not required. The Red wire is for power from 3V to 5V, Black one for Ground (GND) and the Purple wire for Signal processing.

The circuitry of the back of the sensor needs to be protected to get a proper reading from the sensor. It can be easily done by using a hot glue gun.

**2. ADC (Analog to Digital converter) MCP3008:**

Knowing that Raspberry pi doesn’t have any hardware which can convert an analog signal to digital. As we know that heart rate which are pulse rates are analog signals and to convert them into digital signals through Raspberry Pi we are going to use a microchip on our breadboard with the GPIO pins of Raspberry Pi.

The microchip we are going to use which is MCP3008, which includes 10 bit resolution and it has 8 input channels which are programmable.

Facing the chip towards us (according to the Figure 1.1), on the left side there are 8 input channels, pin 1 CH0 to pin 8 CH8 and the right hand side there are,

Pin 16 VDD which is 2.7V to 5.5V power supply where we will use 3.3V power supply from Raspberry Pi,

Pin 15 VREF which is reference voltage output,

Pin14 AGND which is analog ground,

Pin 13 CLK which is serial clock and it maintains minimum clock speed, the device converts one bit of each clock that it received.

Pin 12 Dout which is **Serial Data out,** this Serial peripheral interface (SPI) data out pin normally uses for to shift out the analog to digital conversion.

Pin 11 Din which is **Serial data in**, and its typically uses for load the channel configuration data into the device,

Pin 10 CS/SHDN which chip select/Shut down input.

Connecting those pins can be easily done by using a breadboard, and a few breadboard wire such as male to male jumpers. But to connect with raspberry Pi we will be needed some male to female jumpers.



Figure : A 10 bit MCP3008 pins layout

**SOFTWARE REQUIREMENTS:**

To work with Raspberry Pi, we have a few limitations as we have to download all the requirement software manually, and unpack them and install them into our Pi to run the desired application. The applications or software we needed are:

* Python itself, including its IDLE and a few libraries of it such as RPi.GPIO and wiringpi.

**TOOLS REQUIRED:**

A breadboard, a few male to male, male to female jumper cables, breadboard connectors and a GPIO cable to connect the breadboard with RPi.

**IMPLEMENTATION:**

ECG (echocardiography) is a system where we can check if the heart is beating normal, higher or lower than its ranges. Typically in medical science we use it for patient, but nowadays it is been using vastly in sports and other activities.

In our project we are going to build an ECG which will give us approximate heart rate and there on we can range them as normal or higher according to our choice.

**Step 1:**

Our very first step will be setting up the Raspberry pi and then pulse sensor and of course the ADC microchip MCP3008 onto a breadboard with the help of breadboard jumpers.

To setup the Raspberry pi is very simple as it comes with NOOBS (new out of box software) which includes booting image and operating systems. We install the operating system and we are good to go. For this project we are going to install RasBian which is a Linux based Debian and Raspberry combined operating system. We can either choose GUI based Desktop or command line as well. But we will use the python IDLE (GUI format) for our project.

**Step 2:**

Connecting the Pi with the ADC chip is pretty simple. To do that at first we need to wire up the ADC chip to the breadboard.

1. Pin 16 VDD from the ADC with the pin 1 of Pi which is 3.3V

2. Pin 15 VREF of ADC chip with pin 1 of Pi which is also 3.3V

3. Pin 14 AGND of ADC chip connect with pin 6 of Pi which is GND of Pi as well

4. Pin 13 CLK of ADC chip with pin 23 of Pi

5. Pin 12 Dout of ADC chip with pin 21 of Pi

6. Pin 11 DIN of ADC chip with pin 19 of Pi

7. Pin 10 CS of ADC chip with pin 24 of pi

8. Pin 9 DHND of ADC chip will connect with the Pi pin 6 which is GND of Pi as well.

**Step 3:**

Connecting the Pulse sensor with both ADC and the Pi is very simple. As we mentioned earlier that the Pulse Sensor Amped module comes with colour coded wires, RED, BLACK and PURPLE.

1. The RED wire connects with the pin 1 of Rpi’s 3.3V power supply.

2. Then the PURPLE wire connects with ADC chip’s pin 1 CH0 for analog signal input.

3. And the last wire the BLACK one will connect to GPIO pin 3 through breadboard for GND.

Figure 1.2 shows the wiring diagram with Pi and ADC and the pulse sensor.

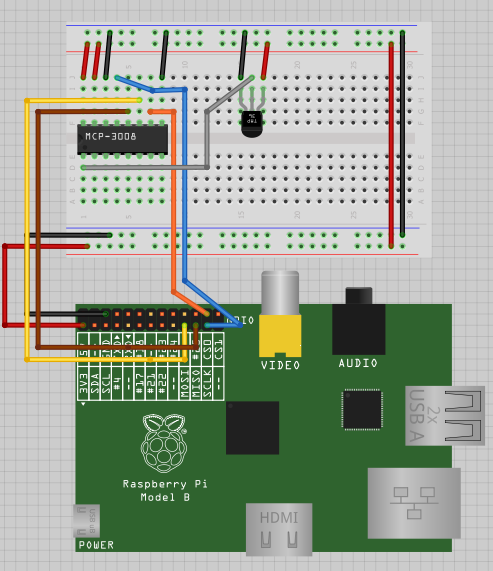
****

Figure : Wiring up Diagram between Pi and ADC

**Step 4:**

After wiring up our hardware we are now going to move on with our software installing and testing part. As we are going to use python codes for our ECG project at first we need to have Python itself on the Pi. Then we are going to install a specific library from python, it’s called RPi.GPIO which will help us to communicate the hardware we setup earlier with the Pi.

To install Python on Pi we need to write this specific code on the command line by clicking on LNX Terminal Folder.

**$ *sudo apt-get update*** // this is to ensure that our Pi is up to date.

**$ *sudo apt-get install python-dev***

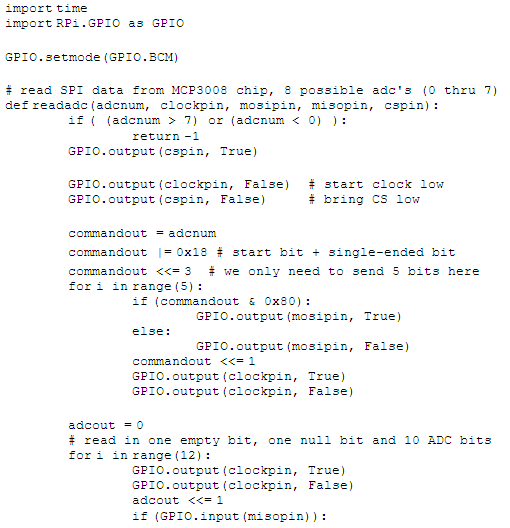
Then follow the instruction and python will be installed in our system. To check that python has been installed we need to write the following command.

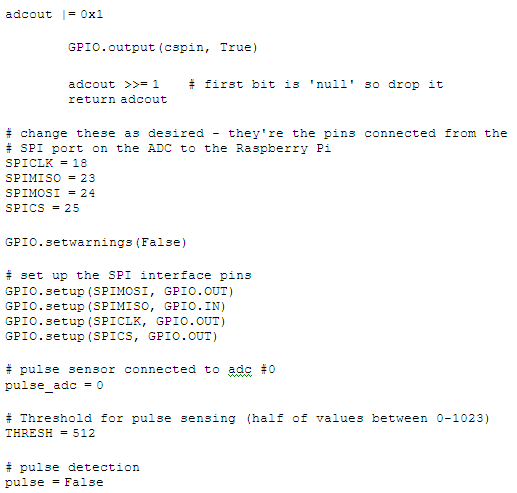
**$ *sudo ls*** // here ‘LS’ means the list of program we have on the Pi.

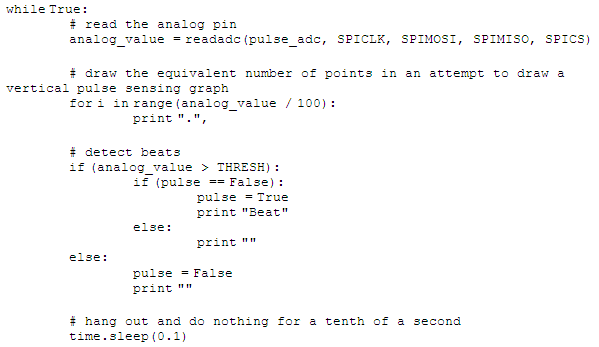
We should get the python installed confirmation after this command regarding its version.

**Step 5:**

Now we need to talk with the hardware and the software we have installed. We need to write some relevant code which will convert our analog pulse signals into digital signals. Codes are written in Python language (as Raspberry Pi technically supports straight with python) and using two of its library, time and RPi.GPIO as we installed earlier. Codes for our pulse reading and converting into a graphical interface are given bellow.



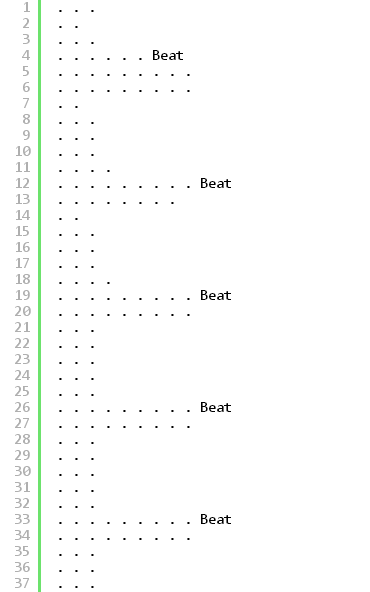




**Testing the Project:**

The most significant part of this project is testing the project is up and running. Saving the python code naming it ECG.py, then we are going to strap the pulse sensor on to an index finger firmly. We need to be careful about wrapping the strap on the finger as we might destroy the module circuits.

After running the ECG.py file with the sensor strapped on the finger we will see something like this:



The display shows that the beat per pulse and highest beat indicate the **Beat** written at the end of the pulse. The loop continues until we disconnect the pulse sensor or end the programme.

It’s a simple Echocardiography system we have made out of Raspberry Pi and with the help of some embedded device and a pulse sensor which is also an embedded module.

**Social, economical and marketing issues:**

In medical science we all know that how important to monitor heart beat (rate) of a patient in terms of treatment or research. Blood pressure measurement is a significant example of pulse sensing.

We have made ourselves a mini ECG machine which is technically not costly at all. A normal ECG machine will cost us approximately GBP300. Here we have made one ECG machine less than GBP100.

By connecting a low price LCD screen to our mini ECG system we can even go ahead for production of a low cost ECG machine which can be provided to the rural area medical centres where the ECG machine is not available or very expensive to afford one.

**Extensions of the project:**

Considering the graphical limitations of applications we can use in our Raspberry Pi, as we can see the graphics of our project is not that great, in this case to improve the graphics of the heart beat we can send the raw pulse signals through some sort of web server or using sockets to another computer to see the improved graphics of our heart beat. We have to make an application using Java or any OO programming language to convert the raw signal into a high quality graphical display. We can even add an additional sensor, such as body temperature sensor to our project to make it more informative.

**REFERENCES:**

**1. MCP3008 microchip pins layout (Figure 1)**

**Source:** <http://learn.adafruit.com/reading-a-analog-in-and-controlling-audio-volume-with-the-raspberry-pi/>