**Abstract**

*This paper describes design and development details of hardware and software components required for testing and verification of PWM using Samsung S3C6410 ARM-11 board. To achieve this, we connect prototype board to the development board. The prototype board has a voltage divider and an op-amp stage soldered to it. An I2C sensor will be mounted on a servo motor. The output will be driven by a PWM output signal. The aim is to perform 1D convolution on the data received from the sensor.*

**Introduction**

Samsung TINY6410 is an ARM-11 based development board. It is powered with Samsung S3C6410 SoC. It is capable of running full fledge operating system. it has got different types of I/O ports like IR, USB, Audio, Ethernet, Camera, TVout, 40 pin system bus, 30 pin GPIO, 20 pin SDIO, etc. It runs Linux operating system. We are going to use GPIO pins to interface with the external prototype board. Software program running on ARM board has two components. Its main driver that actually interfaces with ADC pin runs in kernel mode. User level application program continuously reads the digital value of the voltage connected to PWM pin. We calculate the P(proportional), I(Integral) and D(Differential) based on what we give to the PWM signal to buzzer. Basic components involved in the design are computer, ARM-11 board, prototype board and power adapter. Computer is connected to ARM-11 board using USB to serial connecter. ARM-11 board is connected to LSM303 board.

**Technical Challenges**

The technical challenges involved in this project are both software and hardware in nature. On the hardware side we have to establish a connection between the development board and prototype board. This was quite demanding as we had to ensure that both the ground of the prototype board and development board were connected together, to avoid the CON1 pin from being damaged. On the software side the main challenge was to configure the data and control registers properly to ensure smooth connection and communication. The application program and driver program has to be linked. There were a few issues involved in this. We also had to make sure that the modules are properly compiled.

**Figure 1: - SYSTEM ARCHITECTURE BLOCK DIAGRAM**

**Objectives**

1.To write a driver program and an application program for the ARM-11,CPU.  
  
2.To calculate the PID values and apply it to the PWM pins.

**Requirements**

Considering the design objectives in the previous section the requirement for this experiment are divided in into both hardware and software.

The hardware requirements are as follows.

1. Samsung ARM 11 Development board.  
2. USB to Serial Converter and cable.  
3. Wire wrapping prototype board.

For the software requirement, we need the following:

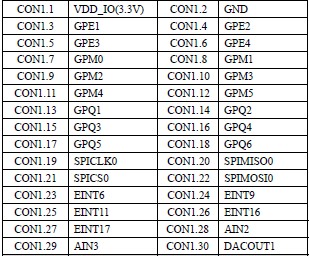
1. Linux Ubuntu.  
2. ARM Linux toolchain.  
3. Putty.

**SAMSUNG ARM-11 BOARD**

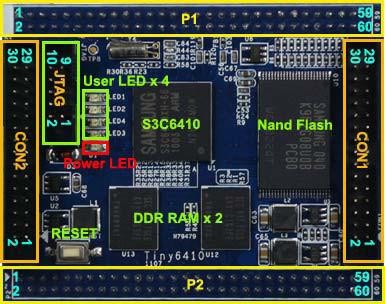
Samsung ARM-11 Board is powered by Samsung S3C6410 SoC. This board has 30 pin GPIO port. Out of which we are going to use 2 pins viz., ADC (Pin28) and GND (Pin2), for our experiment. ADC pin can read voltage values between 0V to 3.3V DC. GND is connected to common ground. This board runs Linux operating system on it. So to interface with ADC pins, we need to write a device driver module. Section 4 explains no how to write device driver software for the same and user level application program to read and validate ADC data. However, in this section, hardware details of Samsung S3C6410 ARM-11 board are explained. Figure 4 shows the picture of that board. And Figure 5 shows picture of CPU module on that board. As you can see in Figure 5, it has two different types of connectors mounted on it, viz., CON1 and CON2. We are going to use CON1 as it has ADC pins of our interest. CPU module on this board is detachable, and detailed picture of CPU module is shown in Figure 5.



**Figure 4. Samsung S3C6410 ARM-11 Board**



**Table 2. Pin Configuration for CON1 Port on CPU Board**



**Figure 5. S3C6410 CPU Module**

|  |  |  |  |
| --- | --- | --- | --- |
| P1.1 | DC-5V | P1.31 | USB Slave D- |
| P1.2 | GND | P1.32 | USB Host D- |
| P1.3 | LCD\_R5/GPJ7 | P1.33 | USB Slave D+ |
| P1.4 | LCD\_R4/GPJ6 | P1.34 | USB Host D+ |
| P1.5 | LCD\_R3/GPJ5 | P1.35 | TSXP/AIN7 |
| P1.6 | LCD\_R2/GPJ4 | P1.36 | TSXM/AIN6 |
| P1.7 | LCD\_R1/GPJ3 | P1.37 | TSYP/AIN5 |
| P1.8 | LCD\_R0/GPJ2 | P1.38 | TSYM/AIN4 |
| P1.9 | LCD\_G5/GPI15 | P1.39 | AIN0 |
| P1.10 | LCD\_G4/GPI14 | P1.40 | AIN1 |
| P1.11 | LCD\_G3/GPI13 | P1.41 | WiFi\_IO/GPP10 |
| P1.12 | LCD\_G2/GPI12 | P1.42 | WiFi\_PD/GPP11 |
| P1.13 | LCD\_G1/GPI11 | P1.43 | SD1\_CLK/GPH0 |
| P1.14 | LCD\_G0/GPI10 | P1.44 | SD1\_CMD/GPH1 |
| P1.15 | LCD\_B5/GPI7 | P1.45 | SDI\_Ncd/gpn10 |
| P1.16 | LCD\_B4/GPI6 | P1.46 | SD1\_Nwp/gpl14 |
| P1.17 | LCD\_B3/GPI5 | P1.47 | SD1\_DAT0/GPH2 |
| P1.18 | LCD\_B2/GPI4 | P1.48 | SD1\_DAT1/GPH3 |
| P1.19 | LCD\_B1/GPI3 | P1.49 | SD1\_DAT2/GPH4 |
| P1.20 | LCD\_B0/GPI2 | P1.50 | SD1\_DAT3/GPH5 |
| P1.21 | VDEN/GPJ10 | P1.51 | DACOUT0 |
| P1.22 | PWM1/GPF15 | P1.52 | PWM0/GPF14 |
| P1.23 | LCD/GPJ9 | P1.53 | XEINT0/GPN0 |
| P1.24 | LCD/GPJ8 | P1.54 | XEINT1/GPN1 |
| P1.25 | LCD/GPJ11 | P1.55 | XEINT2/GPN2 |
| P1.26 | GPE0 | P1.56 | XEINT3/GPN3 |
| P1.27 | VBUS | P1.57 | XEINT4/GPN4 |
| P1.28 | OTGDRV\_VBUS | P1.58 | XEINT5/GPN5 |
| P1.29 | OTGID | P1.59 | XEINT19/GPL11 |
| P1.30 | EINT8/GPN8 | P1.60 | XEINT20/GPL12 |

**Table 3. Pin Configuration for P1 Port on CPU Board**

**HARDWARE DESIGN**

Hardware design has three major components, viz., LSM303, Servo Motor and ARM-11 Board. This section explains implementation details of prototype board and interfacing PWM pin on ARM-11 Board.

**SOFTWARE DESIGN**

Software implementation required for this experiment is divided into two parts, viz., Kernel Driver Module and User Application Program, ARM-11 board runs Linux Operating System on it. So user level applications cannot access hardware directly. Hardware access is allowed to programs running in kernel mode only. So to gain the hardware access, we need to write a kernel device driver module for interfacing with PWM pins. And user level application communicates with this kernel driver via system calls like open, read, etc

There are two different ADC channels on CON1, channel 2 on pin 28 and channel 3 on pin 29. We are going to use pin 28 for our experiment. Pin 2 which is GND pin, is connected to GND on prototype board. ADC on board can read analog voltage on this ADC pin and can convert it to appropriate digital value. FFT and power spectrum is calculated for retrieved values. After that we calculate P.I.D. values, and with help of PWM we sound the buzzer. Software part is explained in detail in the Section 4. This pin configuration for CON1 connector is given in the Table 2.

So there would be just two wires from ARM-11 Board connected to prototype board. ADC i.e. Pin 28 connected to op-amp output, and GND i.e. Pin 2 connected to GND on prototype board. This will complete the hardware implementation of the experiments. Next section i.e. Section 4 describes software part required for the experiment.