AAKASH INTERNSHIP

A PROJECT REPORT

Submitted By

ANAMIKA SHARMA

(108112010)

of

Electronics and Communication Engineering

National Institute of Technology, Tiruchirapalli

Tamil Nadu – 620015

Under the guidance of

Dr. B.Venkataramani

Dr. S.R.Balasundaram

Project coordinators

Dec 2013 – Jan 2014

ACKNOWLEDGEMENT

An Internship is a golden opportunity for learning and self-development. I consider myself very lucky and honoured to have so many wonderful people help us throughout the course of this project.

I express my warm gratitude to Dr. B Venkataramani for selecting me as a part of the Aakash Internship, constantly motivating for doing better and showing complete confidence in my work.

I am indebted to Dr. S.R.Balasundaram for his constant help and support in light of this internship. I am overwhelmed to acknowledge his dedication and humbleness.

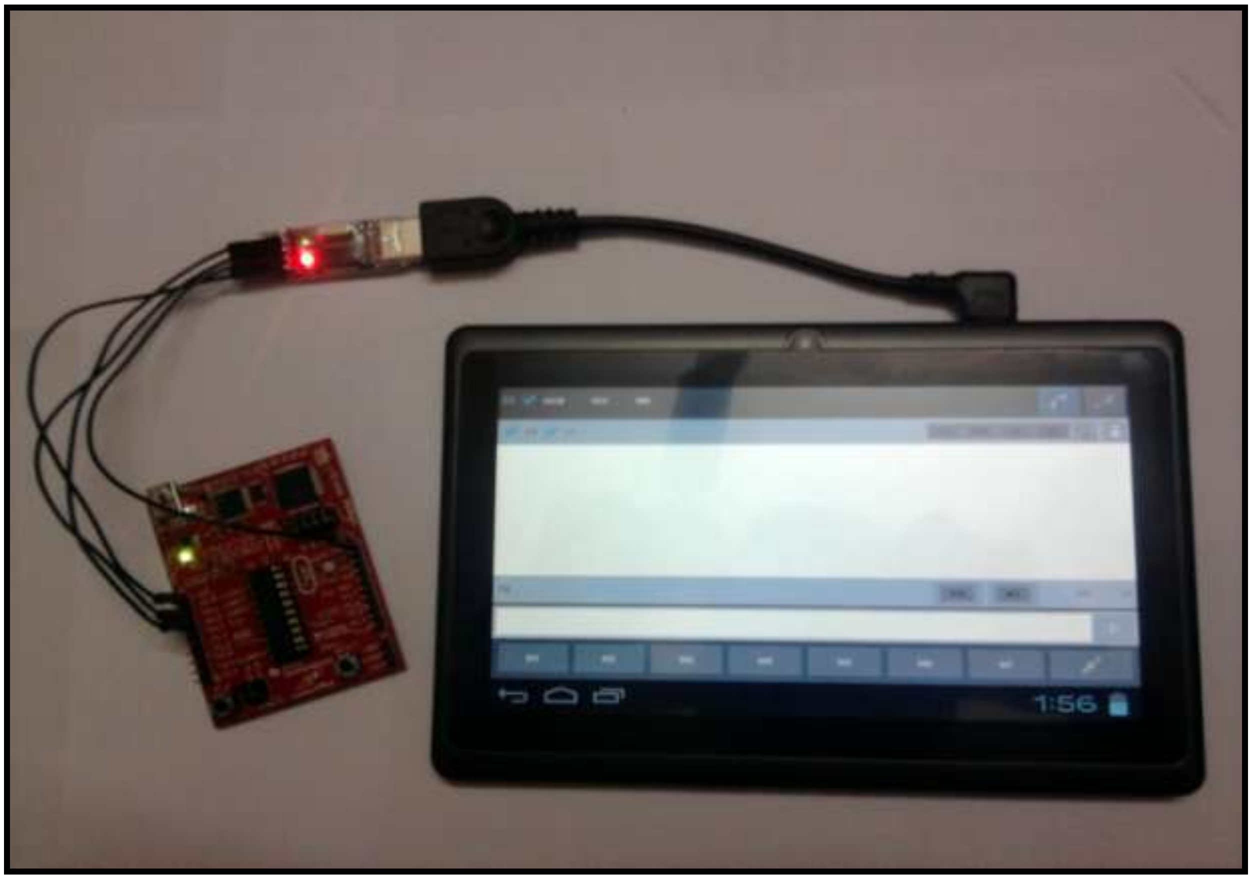
Finally, I would also like to thank all other colleagues working in different projects especially Reishi Kumar, under Aakash Internship for helping me at small problems as well as critical junctures.

PROJECT OVERVIEW

The internship required to work on the design of embedded systems using Aakash tablets. We were expected to learn Android/Linux and interface sensors to the Aakash.

The aim of the project is to interface a Temperature Sensor to Aakash Tablet. The internal Temperature sensor of MSP430 is used for the pupose. The Sensor Data is sent to the Aakash Tablet which has an Android Application installed in it to display the data.

The project also includes controlling the movement of an Arduino bot using the Aakash Tablet interfaced via a USB Cable and controlling the glow of the the onboard red and green leds.



- Interfacing Temperature sensor

CONTENTS

1. INTRODUCTION

1.1. Aakash Tablets in Education……………………………………………………6

1.2. Embedded Systems……………………………………………………………8

1.3. Transducers……………………………………………………………………9

1.4. DACs/ADCs……………………………………………………………………10

1.5. Microprocessors / Microcontroller………………………………10

2. SERIAL COMMUNICATION……………………………………………………12

3. UART………………………………………………………………………………………12

4. TEMPERATURE MONITORING DESIGN

4.1. Block Diagram…………………………………………………………………14

4.2. The Microcontroller………………………………………………………14

4.3. PL2303 Module…………………………………………………………………16

4.4. USB OTG…………………………………………………………………………18

5. SOFTWARE DESIGN……………………………………………………………19

6. ANDROID APPLICATION……………………………………………………23

7. BOT CONTROL

7.1. Block Diagram…………………………………………………………………24

7.2. L293DNE Motor Driver…………………………………………………25

7.3. Arduino Atmega Board…………………………………………………25

7.4. Arduino Source code and Android app output screen……………………………………………………………………………………27

8. LED CONTROL……………………………………………………………………………29

9. SUMMARY AND CONCLUSION……………………………………………31

10. REFERENCES………………………………………………………………………32

INTRODUCTION

AAKASH TABLETS IN EDUCATION:

IIT Bombay along with the Ministry of Human Resource Development has developed Aakash tablets for educational purpose. Aakash 2.0 is a complete educational purpose tablet and not designed with hi-fi configurations. Aakash 2.0 includes 3D-modelling, C++ programming, remote and collaborative training applications, robotic control and live assessment tools.

Aakash is now available in an enhanced version (Aakash-2) of the original tablet with 7 inch multi-touch 4-point capacitive display and 800x480 pixel resolution. It has a 1 GHz processor, 512 MB memory, 4GB portioned NAND flash and micoSD card slot which can extend upto 32 GB memory and it runs the Android 4.0 Operating System.

Aakash 2.0 can be operated with Wi-Fi which enables the students with easy internet connections. It is also equipped with a micro-USB slot in which one can use internet modems provided by different network

providers. Aakash 2.0 is not equipped with third generation facilities. So, one cannot use this tablet for video calling or high speed internet. Aakash 2.0 is featured in such a way that it is best suited for educational purpose, not for commercial use.

Aakash Applications

The tablet comes with variety of applications such blender, clicker and proximity.

a) Blender: Blender is a free and Open Source software product, used for creating animations, rendering, video editing, etc.
b) Clicker: Clicker devices are used to collect instant feedback from a large number of students, either as a response to a question, or to a quiz. A quiz question now gets fully downloaded on individual student's Aakash tablet, through Wi-Fi. A multiple question test can now be conducted. All the questions of such a test are downloaded on the student tablet. Time control is maintained by Aakash. At the end of the test time, all answers are automatically collected, and individual scores get recorded in the back-end system.
c) Proximity: proxyMITY enables creation of interactive lessons, by importing lecture video and presentation slides. The name stands for Proxy Multimedia Integration Tool for You. The entire lesson can be published in the form of either a desktop standalone application, or as html content to be viewed within a web browser.

Tools and Languages

Aakash tablets support C, C++, Python for programming activities. Also, SciLab is available for numerical computations and for research activities.

EMBEDDED SYSTEMS

An embedded system is a computer system designed to perform a specific task within a larger mechanical/electrical system rather than a general purpose computer system for multiple tasks. Embedded system controls many device in common use today. Some examples of such systems are automobile control systems; industrial processes control systems, mobile phones, or small sensor controllers.

Embedded system range from no user interface at all-dedicated only to one task, to complex gui's that resemble modern computer desktop operating system.

Embedded systems talk with the outer world via peripherals, such as:

- Serial Communication Interface(SCI): RS-232
- SSCI
- Universal Serial Bus

Embedded System designers use compilers, assemblers and debuggers to develop embedded systems: More specific tools

- In circuit debuggers/ emulators
- Utilities to checksum or CRC to a program.

Cyclic Redundancy Check (CRC): is an error detecting code commonly used in digital networks and storage devices to detect accidental changes of raw data.

- ➢ An embedded system requires Transducers, amplifiers, DAC/ADC, microprocessor/microcontroller for processing.

Types of Languages that can be used:

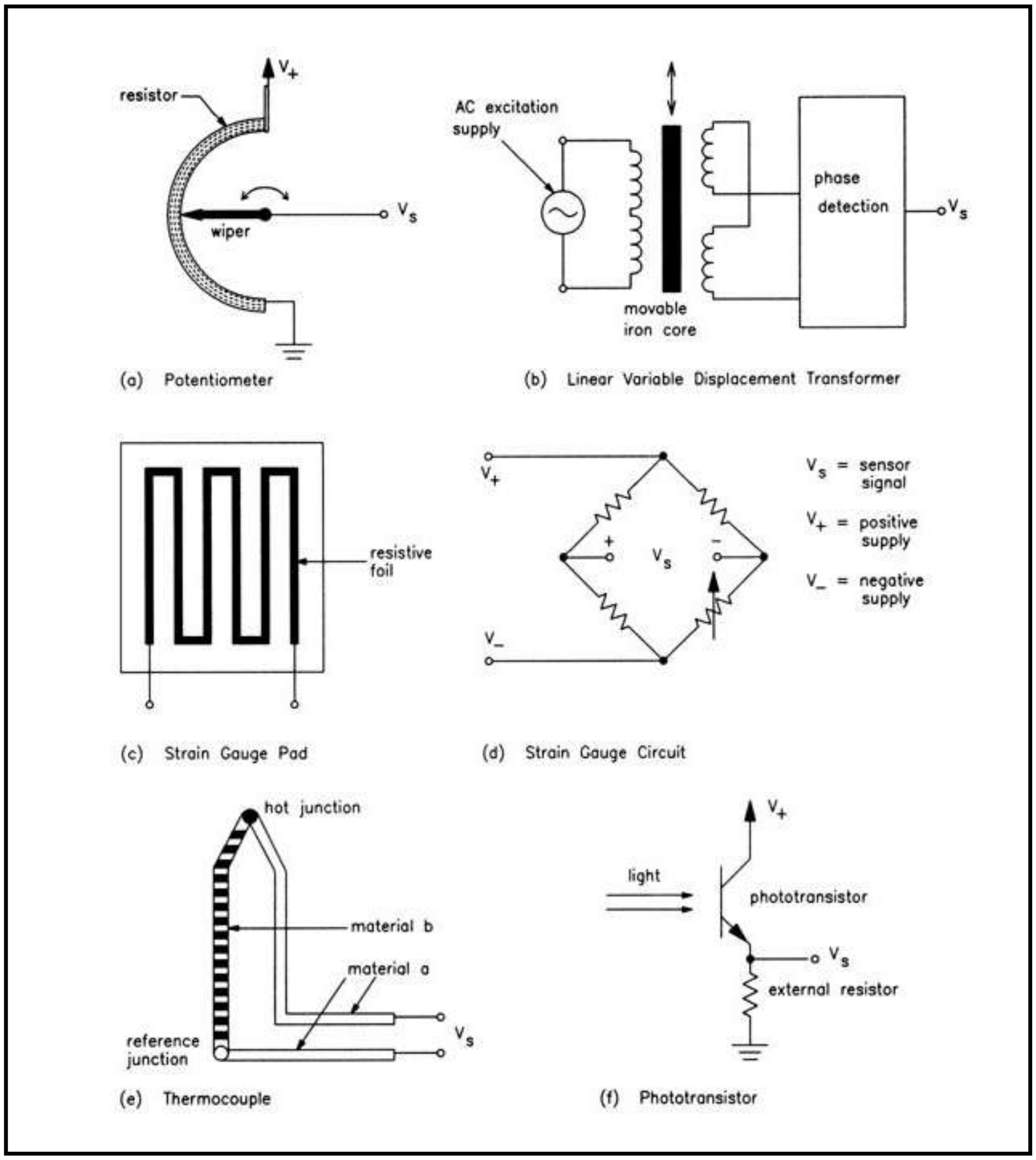
- ➢ Low level Language - Assembly Language
- ➢ High level Language – BASIC , C , C++, Java, PASCAL

High level languages are mostly used because of easier coding and easily portable

TRANSDUCERS: A device that converts a process variable (ex. Car speed) into an electrical signal or vice versa.

Sensors (Input transducers): Potentiometer (position); strain gauge, piezoelectric device (force); thermistor, thermocouple (temperature); photoconductive cell, phototransistor (light); current transformer, SENSEFET (current); microphone (sound), etc.

Actuators (Output transducers): solenoids, relays, speakers; darlington transistors, triacs, etc.



ADCs/DACs

Analog refers to physical quantities that vary continuously instead of discretely. Physical phenomena typically involve analog signals. Examples include temperature, speed, position, pressure, voltage, altitude, etc. Microprocessors work with digital quantities (values taken from the discrete domain).

For a digital system to interact with analog systems, conversion between analog and digital values is needed. Building blocks to perform the conversions are: (1) **Digital to analog converters (DACs),**
(2) **Analog to digital converters (ADCs).** A digital to analog converter has a digital input that specifies an output whose value changes in steps. These step changes are in volts or amperes. The analog to digital converter has an input that can vary from a minimum to a maximum value of volts or amperes. The output is a digital number that represents the input value.

MICROPROCESSORS/MICROCONTROLLERS

Microprocessor: The microprocessor is a programmable device that takes in numbers, performs on them arithmetic or logical operations according to the program stored in memory and then produces other numbers as a result.

- Microprocessor is a silicon chip which includes ALU, register circuits & control circuits.
- Microcontroller is a silicon chip which includes microprocessor, memory & Input, Output in a single package.

Microcontroller: A microcontroller is a device that performs a series or a sequence of instructions as given by the user. These instructions (program) are stored in the memory and executed by the device. It has input and output devices interfaced with it.

Microcontrollers are designed for embedded applications, in contrast to the microprocessors used in personal computers or other general purpose applications.

General Microcontrollers available in the market:

- TI MSP430
- Atmel AVR Atmega 8,16,32
- PIC
- Cortex

Internal Architecture:

- **Von Neumann-**Data and Memory is accessed by the processor in a single bus e.g. MSP430

- **Harvard-** Data and Memory is accessed by the processor in separate buses e.g. Atmel, etc.

Embedded System programming: Platforms used for embedded system programming:

- Microprocessors/Microcontrollers
- FGPAs: Field Programmable Gate Arrays
- Programmable DSPs
- ASICs: Application Specific Integrated Circuits
- Tablets (Future Potential)

SERIAL COMMUNICATION

RS232 is an asynchronous serial communication protocol widely used in computers and digital systems. Asynchronous because there is no separate synchronizing clock signal as there are in other serial protocols like SPI and I2C. The protocol is such that it automatically synchronizes itself. We can use RS232 to easily create a data link between our MCU based projects and standard PC.

UART - UNIVERSAL **ASYNCHRONOUS** RECEIVER TRANSMITTER

A UART is an individual (or a part of an) integrated circuit used for serial communications over a computer or peripheral device serial port. UARTs are included I microcontrollers. The UART takes bytes of data and transmits the individual bits in a sequential fashion. At destination, a second UART re-assembles the bits into complete bytes.

In UART, a start bit and stop bits are used to synchronize the incoming data line. As there is no "clock" line so for synchronization accurate timing is required so transmissions are carried out with certain standard speeds. The speeds are measured in bits per second. Number of bits transmitted is also known as BAUD RATE. Some standard baud rates are 1200, 2400, 4800, 9600, 19200 ... etc. To transmit a single byte we need to extra bits they are START BIT AND STOP BIT. BITS are transmitted as a packet of 7 or 8 bits (ASCII code).

The right-most (least significant) bit is always transmitted first. If parity is present, the parity bit comes after the data bits but before he stop bit(s).

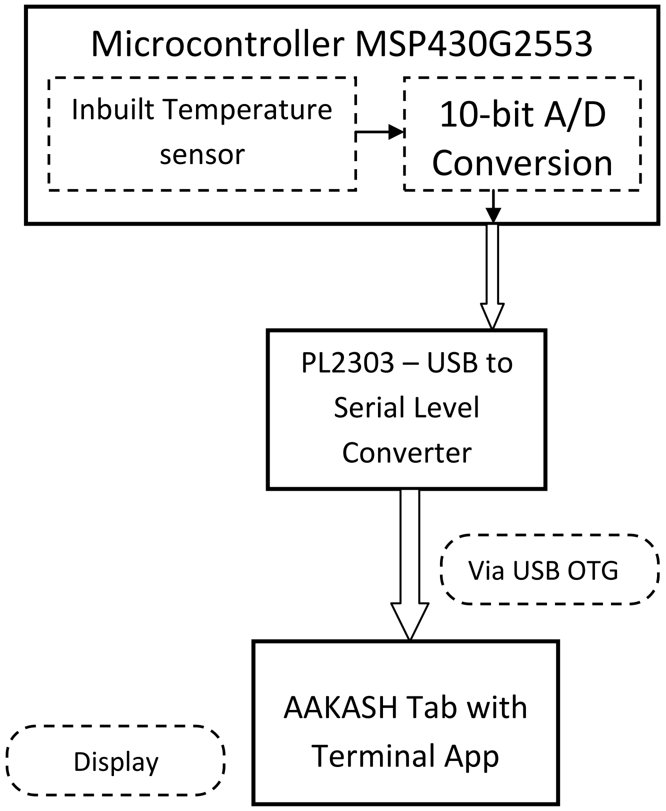
STOP BIT	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	START BIT

UART IN MSP430:

- Half duplex
- Types of UART available :
    1. Hardware
    2. Software
- Receiving and Transmission Functions
- Need for Interrupts and Timers

TEMPERATURE MONITORING DESIGN

BLOCK DIAGRAM:


Microcontroller MSP430G2553
Inbuilt Temperature sensor
10-bit A/D Conversion
PL2303 – USB to Serial Level Converter
Via USB OTG
AAKASH Tab with Terminal App
Display

THE MICROCONTROLLER

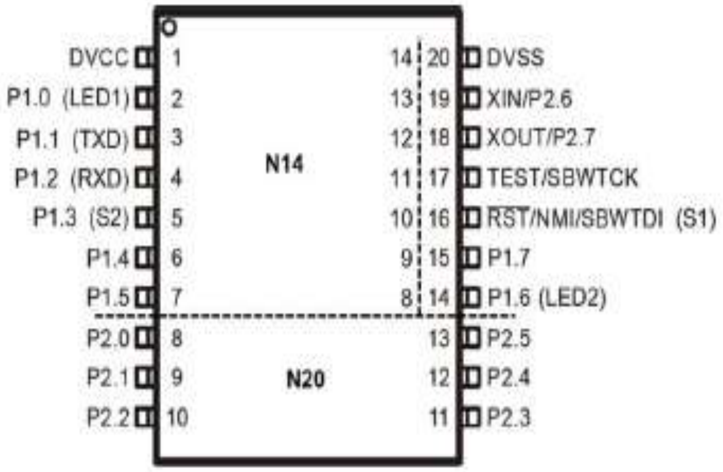
The MSP430 is the family name of the ultra-low power 16-bit mixed-signal RISC processors from Texas Instruments. The 16-bit central processing unit (CPU), the peripherals and the flexible clock system are combined by using Neumann architecture with common memory address bus and memory data bus. Both the address and data buses are 16-bit wide, as well as the registers. They can be used interchangeably for either data or memory addresses. This makes the microcontroller unit (MCU) simpler than any 8-bit processor with 16-bit addresses.

The MSP430 has 16 registers, and the ability to perform arithmetic directly on values in the memory. C compilers can benefit from this and produce more compact, efficient code.

The MSP430 family offers three types of serial communication peripherals:

- Universal Serial Interface (USI)
- Universal Serial Communication Interface (USCI)
- Universal Synchronous/Asynchronous Receiver/Transmitter (USART)

MSP430 Launchpad



Pin Diagram

<u>MSP430G2553 microcontroller</u>: 16-bit Ultra-Low-Power Microcontroller, 16kB Flash, 512B RAM, 10-Bit SAR A/D, Comparator, USCI for I2C/SPI/UART, 24 Touch-Sense Enabled I/O Pins.

The MSP430G2553 microcontroller has an integrated internal temperature. It is possoble to use the internal temperature sensor and obtain a localized temperature reading. Due to the ultra-low-power nature of the MSP430 device, it experiences very little self-heating effect and can therefore give an excellent indication of the surrounding ambient air temperature. Slight device-to-device variations of temperature sensor offset and gain may be present so, depending upon the application, it may be necessary to carry out individual device calibration to remove or reduce these factors.

Need for a level converter

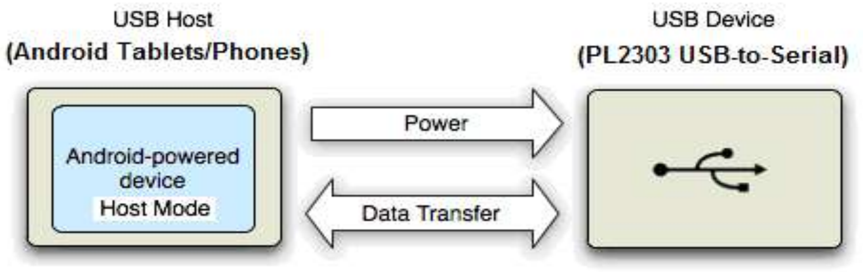
RS 232 LOGIC :	**HIGH: -12V**	
	LOW: +12V	
TTL LOGIC:	**HIGH: 3.3V**	
	LOW: 0V	

So we need an interface to compensate this mismatch.
This interface is provided by an IC called **PL2303**.

Introduction about pl2303 chip: The PL-2303 operates as a bridge between one USB port and one standard RS232 Serial port. The two large on-chip buffers accommodate data flow from two different buses. The USB bulk-type data is adopted for maximum data transfer. Automatic handshake is supported at the Serial port. With these, a much higher baud rate can be achieved compared to the legacy UART controller.
This device is also compliant with USB power management and remote wakeup scheme. Only minimum power is consumed from the host during Suspend. By integrating all the function in a SSOP-28 package, this chip is suitable for cable embedding. Users just simply hook the cable into PC orhub's USB port, and then they can connect to any RS-232 devices.

Pin Assignment of PL2303 Module: TXD (Data output to Serial Port), RXD(Data input from Serial Bus), 3V3(3.3V power for USB transceiver), 5V(5V power), GND(Ground)

PL2303 Android driver Solution: Prolific provides Android USB Host API driver solution (NO root permission needed) for connecting PL2303 USB-to-Serial devices to Android-powered devices with USB Host API mode support. The PL2303 Android JAVA driver library allows customers and developers to write Android applications to interface their PL2303 RS-232/UART devices with millions of Android USB host devices like Smart phones, Tablets, and Smart TVs.



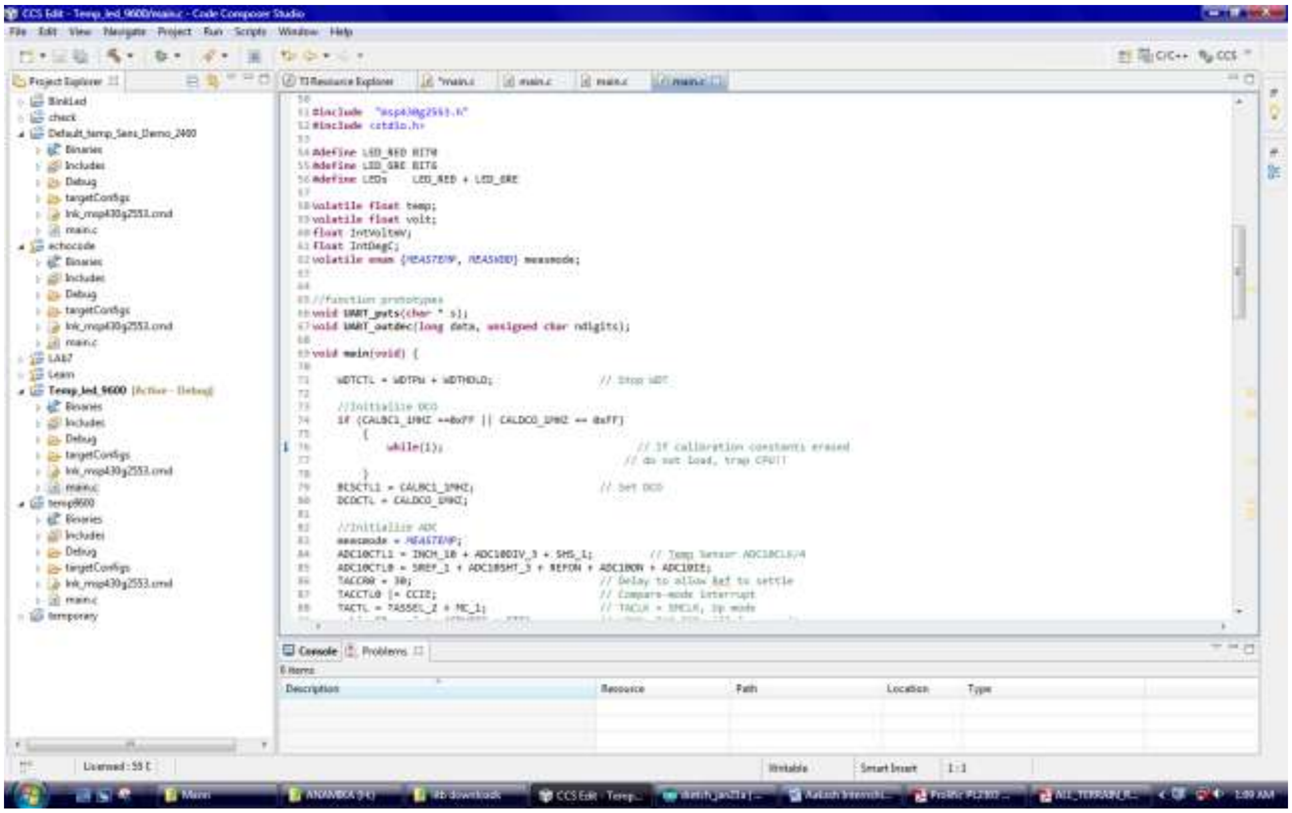
USB On-The-Go (USB OTG)

USB OTG is a specification that allows USB devices such as digital audio players or mobile phones to act as a host, allowing other USB devices like a USB flash drive, digital camera, mouse, or keyboard to be attached to them. Unlike conventional USB systems, USB OTG systems can drop the hosting role and act as normal USB devices when attached to another host. This can be used to allow a mobile phone to act as host for a flash drive and read its contents, downloading music for instance, but then act as a flash drive when plugged into a host computer and allow the host to read data from the device.

The Aakash Tab has got a microUSB port. It is used for transferring files back and forth with the PC. A USB OTG cable with a male microUSB connector on one end and a female full size USB port on the other enables us to interface the PL2303 module to the tab.

SOFTWARE DESIGN

i) Code Composer Studio: The Code Composer Studio is used for programming the codes in C language and for running, debugging and uploading in the microcontroller.



```
#include  "msp430g2553.h"
#include <stdio.h>

#define LED_RED BIT0
#define LED_GRE BIT6
#define LEDs    LED_RED + LED_GRE

volatile float temp;
volatile float volt;
float IntVoltmV;
float IntDegC;
volatile enum {MEASTEMP, MEASVDD} measmode;


//function prototypes
void UART_puts(char * s);
void UART_outdec(long data, unsigned char ndigits);

void main(void) {

    WDTCTL = WDTPW + WDTHOLD;                  // Stop WDT

    //Initialize DCO
    if (CALBC1_1MHZ ==0xFF || CALDCO_1MHZ == 0xFF)
        {
            while(1);                              // If calibration constants erased
                                                 // do not load, trap CPU!!
        }
    BCSCTL1 = CALBC1_1MHZ;                     // Set DCO
    DCOCTL = CALDCO_1MHZ;

    //Initialize ADC
    measmode = MEASTEMP;
    ADC10CTL1 = INCH_10 + ADC10DIV_3 + SHS_1;          // Temp Sensor ADC10CLK/4
    ADC10CTL0 = SREF_1 + ADC10SHT_3 + REFON + ADC10ON + ADC10IE;
    TACCR0 = 30;                               // Delay to allow Ref to settle
    TACCTL0 |= CCIE;                           // Compare-mode interrupt
    TACTL = TASSEL_2 + MC_1;                   // TACLK = SMCLK, Up mode
```

Source Code: The following code measures the Temperature in oC stored in IntDegC (ADC sample is made on A10 with reference to internal 1.5V Vref.) and MSP430 power supply voltage in mV stored in IntVoltmV (ADC sample is made on A11 with reference to internal 2.5V Vref.)

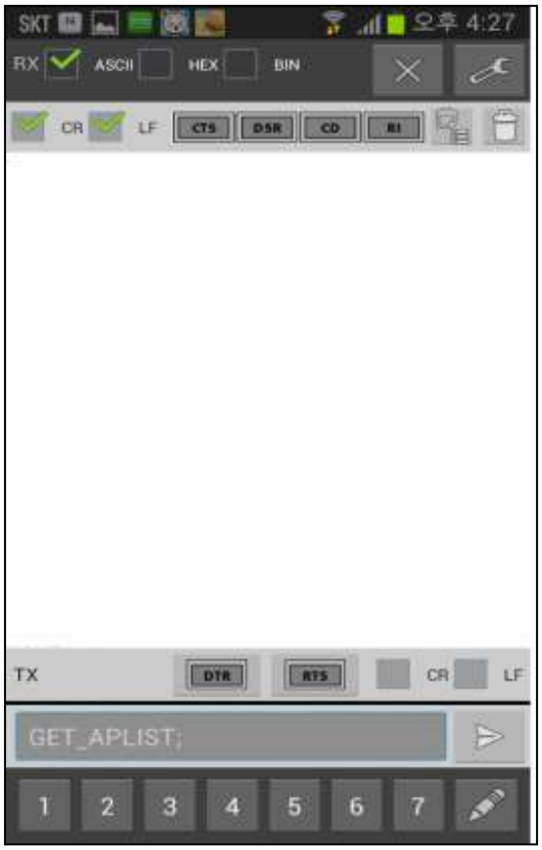
```
#include "msp430g2553.h"
#include <stdio.h>

#define LED_RED     BIT0
#define LED_GRE     BIT6
#define LEDs        LED_RED + LED_GRE
```

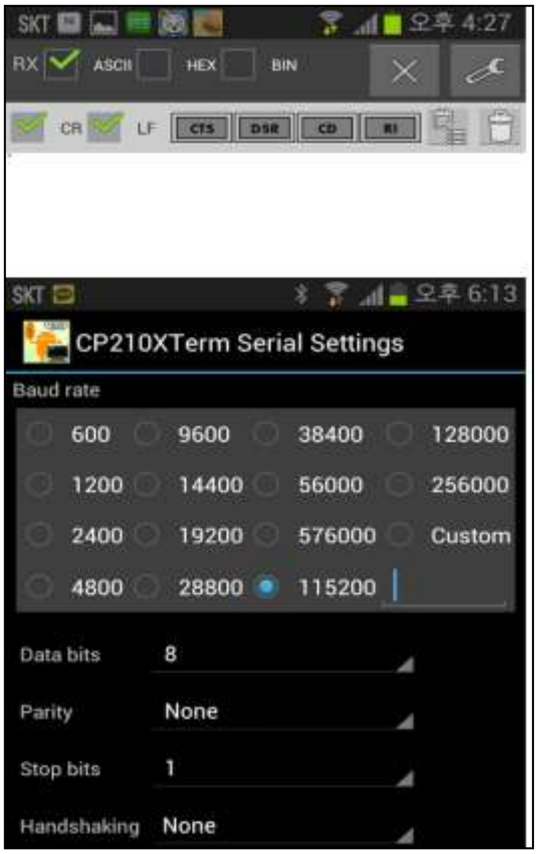
```
volatile float temp;
volatile float volt;
float IntVoltmV;
float IntDegC;
volatile enum {MEASTEMP, MEASVDD} measmode;

//function prototypes
void UART_puts(char * s);
void UART_outdec(long data, unsigned char ndigits);

void main(void) {

     WDTCTL = WDTPW + WDTHOLD;                  // Stop WDT

     //Initialize DCO
     if (CALBC1_1MHZ ==0xFF || CALDCO_1MHZ == 0xFF)
         {
              while(1);        // If calibration constants erased
                                // do not load, trap CPU!!
         }
     BCSCTL1 = CALBC1_1MHZ;          // Set DCO
     DCOCTL = CALDCO_1MHZ;

     //Initialize ADC
     measmode = MEASTEMP;
     ADC10CTL1 = INCH_10 + ADC10DIV_3 + SHS_1;
// Temp Sensor ADC10CLK/4
ADC10CTL0 = SREF_1 + ADC10SHT_3 + REFON + ADC10ON + ADC10IE;
TACCR0 = 30;                       // Delay to allow Ref to settle
TACCTL0 |= CCIE;                   // Compare-mode interrupt
TACTL = TASSEL_2 + MC_1;           // TACLK = SMCLK, Up mode
__bis_SR_register(CPUOFF + GIE);   // LPM0, TA0_ISR will force exit
TACCTL0 &= ~CCIE;                  // Disable timer Interrupt

BCSCTL3 |= LFXT1S_2;               // ACLK = VLO
TACCR0 = 12000;

TACCTL1 = OUTMOD_3;                // TACCR1 set/reset
TACCR1 = 6000;                     // TACCR1 PWM Duty Cycle
     //TACCTL0 |= CCIE;            // Compare-mode interrupt
TACTL = TASSEL_1 + MC_1;      // TACLK = ACLK, Up mode

     //Initialize HW UART
P1SEL = BIT1 + BIT2 ;               // P1.1 = RXD, P1.2=TXD
P1SEL2 = BIT1 + BIT2;               // Secondary peripheral module
function is selected.
UCA0CTL1 |= UCSSEL_2;              // SMCLK
UCA0BR0 = 104;                     // 1MHz 9600
UCA0BR1 = 0;                      // 1MHz 9600
UCA0MCTL = UCBRS0;                // Modulation UCBRSx = 1
```

```
UCA0CTL1 &= ~UCSWRST;              // **Initialize USCI state
machine**

// Initialize LED_RED LED_GRE
P1DIR |= LED_RED + LED_GRE;

     ADC10CTL0 |= ENC;
     while(1) {
          __bis_SR_register(CPUOFF + GIE);      // LPM0 with
interrupts enabled

     switch (measmode) {
                 case MEASTEMP :
                    UART_puts("\nTemp =");
                    P1OUT |= LED_RED;
                    temp = ADC10MEM;
                    IntDegC = (((temp - 673) * 425) / 1024.0)*100;
                    UART_outdec(IntDegC,2);
                    UART_puts(" C");
                    measmode = MEASVDD;
                    break;
                 case MEASVDD:
                    UART_puts(" Volt =");
                    P1OUT |= LED_GRE;
                    volt = ADC10MEM;
                    IntVoltmV = volt * 5000 / 1024.0;
                    UART_outdec(IntVoltmV,0);
                    UART_puts(" mV");
                    measmode = MEASTEMP;
                    break;
          }
P1OUT &= ~(LED_RED + LED_GRE)
 __no_operation(); }
      }

//---------------------------------------------------
// function definitions
//---------------------------------------------------

void UART_puts(char * s) {
     while (*s) {
          while (!(IFG2&UCA0TXIFG));  // USCI_A0 TX buffer ready?
          UCA0TXBUF = *s++;
     }
}


void UART_outdec(long data, unsigned char ndigits){
     unsigned char sign, s[6];
     unsigned int i;
     sign = ' ';
     if(data < 0) {
          sign='-';
          data = -data;
     }
     i = 0;
     do {
          s[i++] = data % 10 + '0';
          if(i == ndigits) {
               s[i++]='.';
          }
     } while( (data /= 10) > 0);
     s[i] = sign;
     do {
          while (!(IFG2&UCA0TXIFG));
          UCA0TXBUF = s[i];
     } while(i--);
}



// ADC10 interrupt service routine
#pragma vector=ADC10_VECTOR
__interrupt void ADC10_ISR(void)
{
  __bic_SR_register_on_exit(CPUOFF); //Clear CPUOFF bit from 0(SR)
     ADC10CTL0 &= ~ENC;                // ADC10 disable
     switch (measmode) {
          case MEASTEMP:
               //temp = ADC10MEM;
               ADC10CTL1 = INCH_11 + ADC10DIV_3 + SHS_1;
               ADC10CTL0 |= REF2_5V;
               break;

          case MEASVDD:
               //volt = ADC10MEM;
               ADC10CTL1 = INCH_10 + ADC10DIV_3 + SHS_1;
               ADC10CTL0 &= ~REF2_5V;
               break;
     }
     ADC10CTL0 |= ENC;}

#pragma vector=TIMER0_A0_VECTOR
__interrupt void TA0_ISR(void)
{
__bic_SR_register_on_exit(CPUOFF); // Clear CPUOFF bit from 0(SR)

}
```

ANDROID APPLICATION

i) FREE USB SERIAL TERM- Android Terminal App from play.google.com is used for displaying the measured temperature. Settings button (wrench icon) of upper right corner is clicked in order to set serial communication. Connect button (phone icon) of upper right corner is clicked in order to connect to the device.





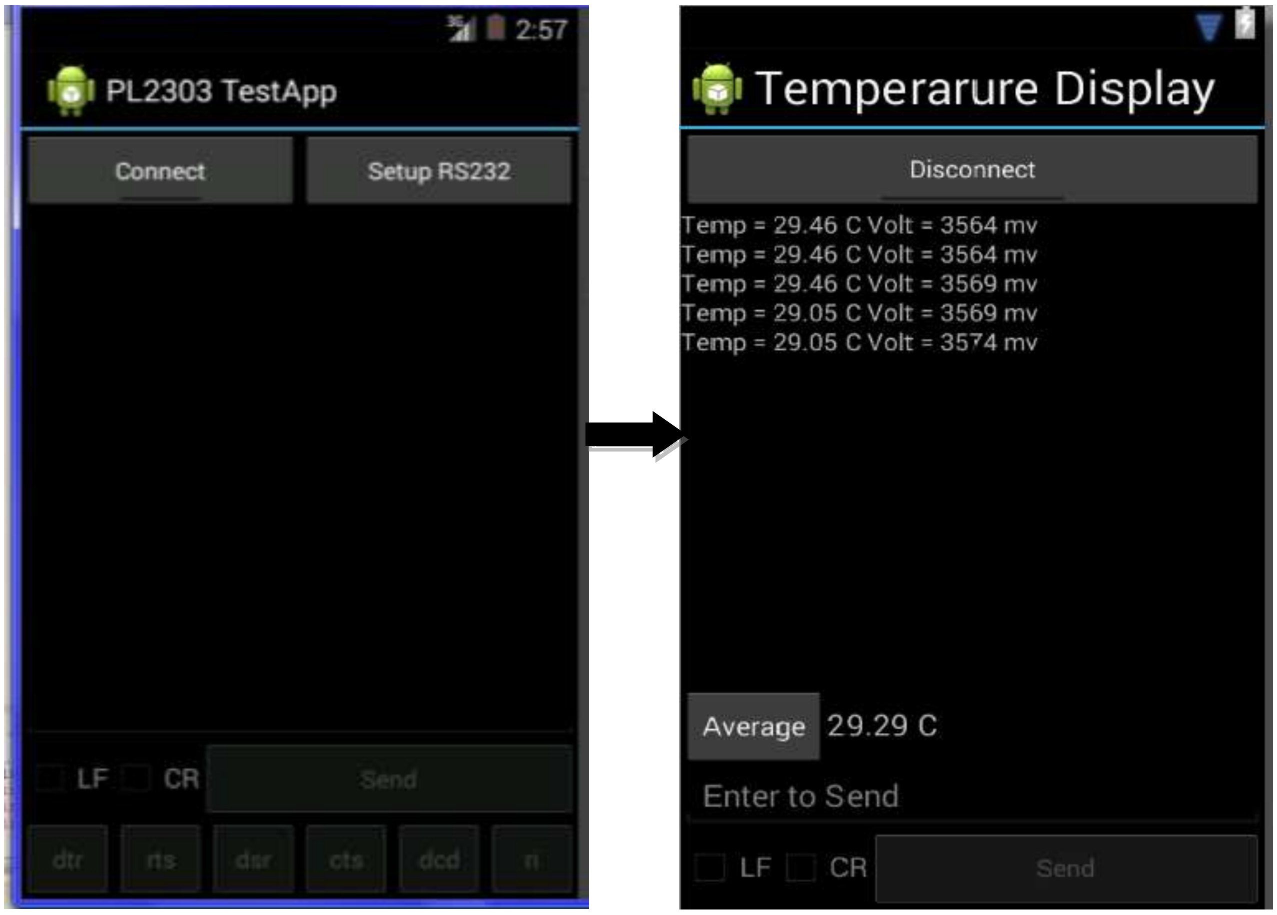
CP210XTerm Serial Settings

Baud rate

○ 600	○ 9600	○ 38400	○ 128000
○ 1200	○ 14400	○ 56000	○ 256000
○ 2400	○ 19200	○ 576000	○ Custom
○ 4800	○ 28800	● 115200	

Data bits	8
Parity	None
Stop bits	1
Handshaking	None

ii) Prolific provides us Pl2303TEST app to display the serially transmitted data from the microcontroller. It displays the Temperature in Celsius and MSP430 power supply voltage in mV.



```
Temp = 29.46 C Volt = 3564 mv
Temp = 29.46 C Volt = 3564 mv
Temp = 29.46 C Volt = 3569 mv
Temp = 29.05 C Volt = 3569 mv
Temp = 29.05 C Volt = 3574 mv
```

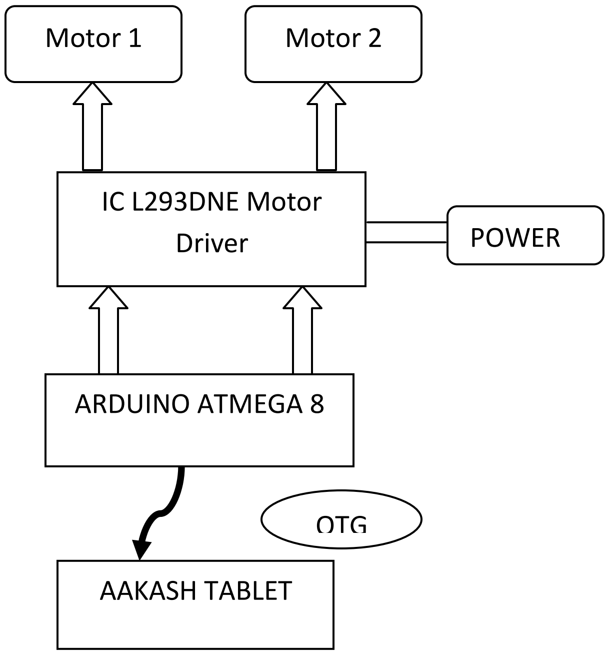
Average 29.29 C

The PL2303TestApp is modified into “Temperature Display” App using Java-ADT application for building Android Apps. It also enables us to calculate the average of the Temperature.

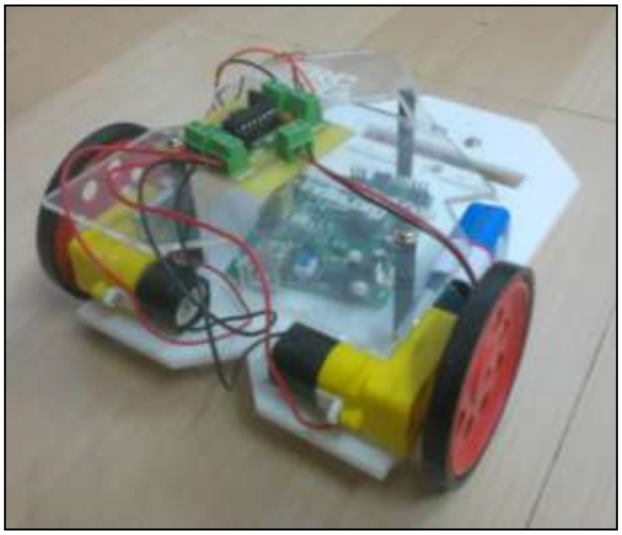
BOT CONTROL

The aim here is to drive a small Arduino bot of the simplest kind with an android app, through USB Cable connection. Arduino board with atmega 8 microcontroller is used for controlling the bot along with the L293DNE motor driver to drive two motors to drive wheels of the bot.

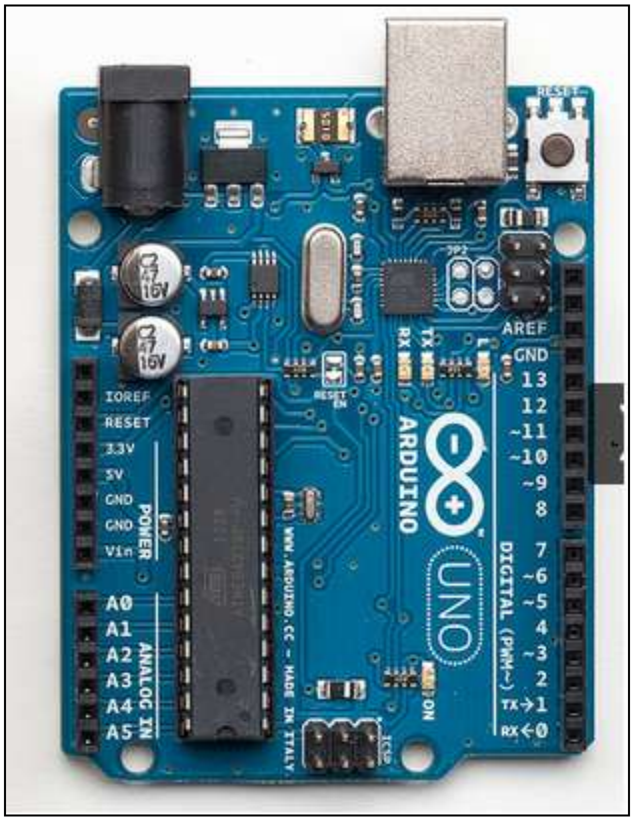
BLOCK DIAGRAM:



L293DNE MOTOR DRIVER



The L293D is quadruple high-current half-H drivers. The L293D is designed to provide bidirectional drive currents of up to 600-mA at voltages from 4.5 V to 36 V. It is designed to drive inductive loads such as relays, solenoids, dc and bipolar stepping motors, as well as other high-current/high-voltage loads in positive-supply applications. All inputs are TTL compatible. Drivers are enabled in pairs. When the enable input is low, those drivers are disabled, and their outputs are off and in the high-impedance state. With the proper data inputs, each pair of drivers forms a full-H (or bridge) reversible drive suitable for solenoid or motor applications.



ARDUINO ATMEGA 8 BOARD

ATMEGA 8: The low-power Atmel 8-bit AVR RISC-based microcontroller combines 8KB of programmable flash memory, 1KB of SRAM, 512K EEPROM, and a 6 or 8 channel 10-bit A/D converter. The device supports throughput of 16 MIPS at 16 MHz and operates between 2.7-5.5 volts.

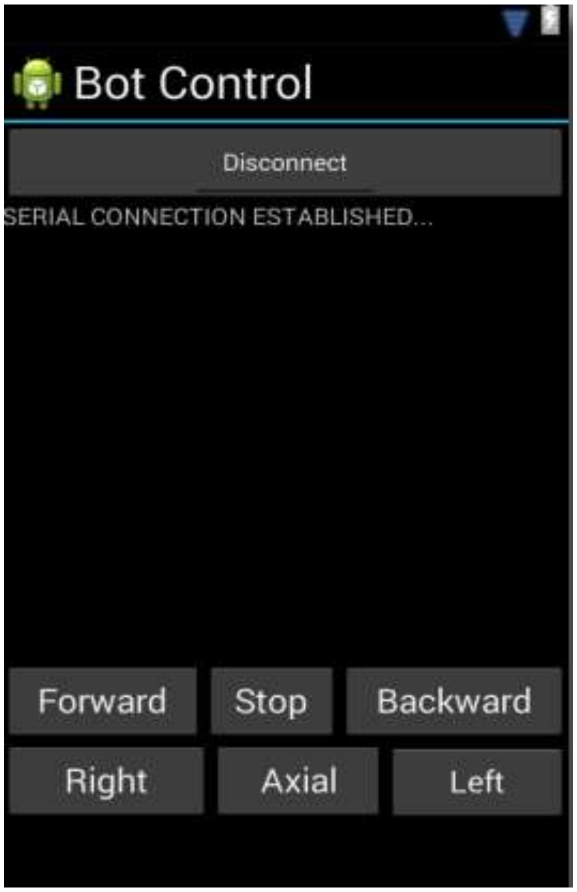
ARDUINO: Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. It's intended for artists, designers, hobbyists, and anyone interested in creating interactive objects or environments.

Arduino can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators. The microcontroller on the board is programmed using the Arduino programming language (based on Wiring) and the Arduino development environment (based on Processing). Arduino projects can be stand-alone or they can communicate with software running on a computer (e.g. Flash, Processing, MaxMSP).

ARDUINO SOURCE CODE

```
char inbyte = 0;         // incoming serial byte
void setup()
{
  // start serial port at 9600 bps:
  Serial.begin(9600);
  pinMode(13,OUTPUT);
  pinMode(12,OUTPUT);
  pinMode(11,OUTPUT);
  pinMode(10,OUTPUT);

  Serial.println("SERIAL COMMUNICATION ESTABLISHED...\n");

}

void loop()
{
  // if we get a valid byte, read analog ins:
  if (Serial.available() > 0) {
    // get incoming byte:
    inbyte = Serial.read();

if (inbyte=='r')
{
  Serial.println("Right Turn");
  digitalWrite(13,LOW);
 digitalWrite(12,HIGH);
 digitalWrite(11,LOW);
 digitalWrite(10,LOW);
}
else if (inbyte == 'l')
{
    Serial.println("Left Turn");
  digitalWrite(13,LOW);
 digitalWrite(12,LOW);
 digitalWrite(11,LOW);
 digitalWrite(10,HIGH);
 }
 else if (inbyte=='f')
 {
     Serial.println("Forward Turn");
   digitalWrite(13,LOW);
 digitalWrite(12,HIGH);
 digitalWrite(11,LOW);
 digitalWrite(10,HIGH);
 }
 else if (inbyte=='b')
 {
     Serial.println("BackWard Movement");
   digitalWrite(13,HIGH);
 digitalWrite(12,LOW);
 digitalWrite(11,HIGH);
 digitalWrite(10,LOW);
 }
 else if(inbyte=='a')
 {
   Serial.println("Axial Turn");
  digitalWrite(13,HIGH);
  digitalWrite(12,LOW);
  digitalWrite(11,LOW);
  digitalWrite(10,HIGH);
 }
else
{
  Serial.println("Stopping... !!!");
  digitalWrite(13,HIGH);
 digitalWrite(12,HIGH);
 digitalWrite(11,HIGH);
 digitalWrite(10,HIGH);
}
  }
}
```

ANDROID APP: OUTPUT SCREEN



SERIAL CONNECTION ESTABLISHED...

LED CONTROL

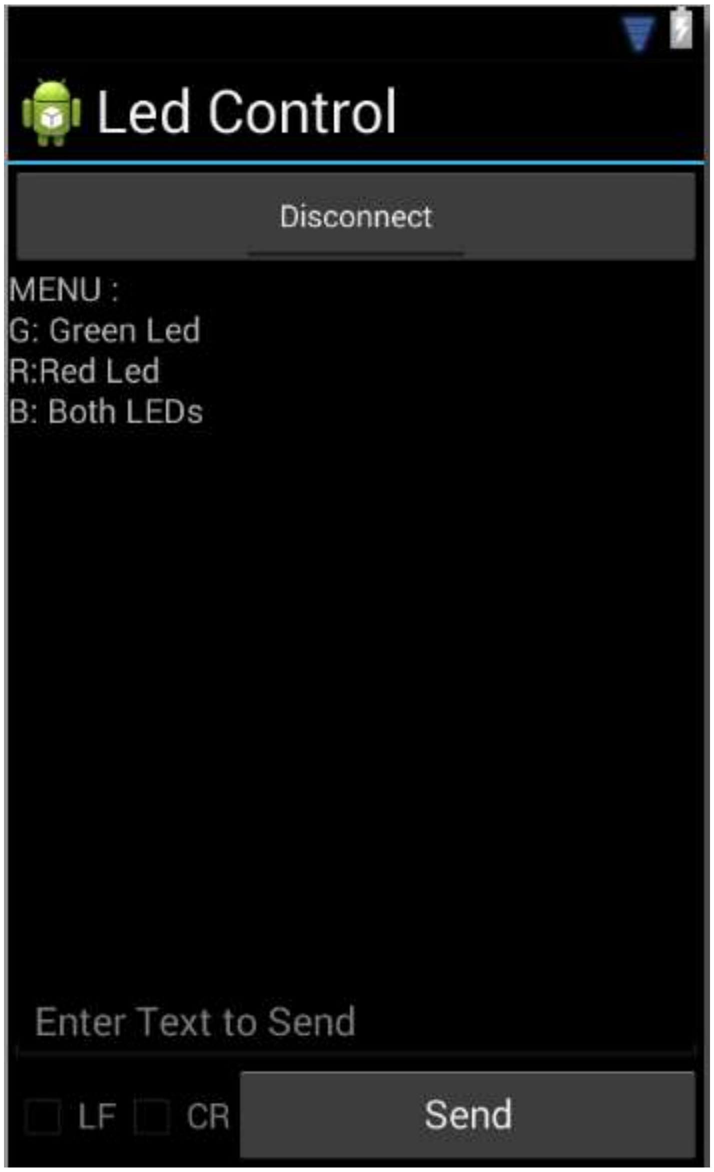
The following project controls the glow of onboard red and green Leds on the MSP430 launch pad depending on the character send.
Energia is used as the platform to compile and upload the code in the microcontroller.

SOURCE CODE

```
void setup()
{
  // setup code: to run once:
  Serial.begin(9600);
  Serial.println("MENU : ");
  Serial.println("G:Green led");
  Serial.println("R:Red led");
  Serial.println("B:Both");
 }

void loop()
{
  if (Serial.available()>0)    // main code: to run repeatedly
 { char inbyte=Serial.read();

 if (inbyte=='B')
 {pinMode(2,HIGH);
  pinMode(14,HIGH);
}
else if (inbyte=='G')
 {pinMode(2,LOW);
 pinMode(14,HIGH);
 }
else if (inbyte=='R')
  {pinMode(14,LOW);
  pinMode(2,HIGH);
  }
else
  { Serial.println("Wront Input\n");
  pinMode(2,LOW);
  pinMode(14,LOW);
}
}
}
```

ANDROID APP: OUTPUT SCREEN

Android App installed on the Tab for the Application:



MENU :
G: Green Led
R:Red Led
B: Both LEDs

SUMMARY and CONCLUSION

The Temperature Sensor has been successfully interfaced to the Aakash Tablet. The Temperature data is obtained by using MSP430 microcontroller having an internal temperature sensor. The Sensor data is then transmitted through Pl2303 level converter to the Aakash Tablet via OTG Cable. Android application named “Free Usb Serial Term” and “Temperature Display” is used for sensor data acquisition and average calculation.

The Arduino bot was successful interfaced with the Aakash Tablet. The movement of the bot could be controlled by Aakash Tablet.

The Aakash Tablet provides a strong platform for developing and implementing microcontroller based embedded projects for the advanced developer and hobbyists.

REFERENCES

- USB On-the-Go- Wikipedia, the free encyclopedia
- Texas instruments website: www.ti.com
- www.google.com
- Android apps on Google Play: https://play.google.com/store/apps/details?id=com.oneman.freeusbtools&hl=en
- Android App Development: http://developer.android.com
- Java Basics for Android Development Part 1&2: http://blog.teamtreehouse.com/java-basics-for-android-development-part-1 http://blog.teamtreehouse.com/java-basics-for-android-development-part-2
- http://stackoverflow.com
- http://www.prolific.com.tw/UserFiles/files/PL2303HXD_Android_SDK_v1009.zip
- http://code.google.com/p/indiserver/downloads/detail?name=PL2303test.zip&can=2&q=label%3AFeatured
- http://www.silabs.com/products/mcu/pages/usbtouartbridgevcpdrivers.aspx