INSTALLATIONS

Installing Apps

1. Install F-Droid.
2. Open f-droid on your mobile and install the following apps:
    - Termux
    - Termux:API

Setting up Termux

1.Give termux access to your user directory in Android.

```
termux-setup-storage
```

2.Upgrade packages (any **one** command)

```
pkg upg
```

- `apt update && apt update`

```
3.Install mandatory packages (any one command)
```

```
apt install build-essential openssh curl git wget subversion
silversearcher-ag imagemagick proot proot-distro python bsdtar mutt nmap neovim
```

```
pkg in build-essential openssh curl git wget subversion silversearcher-ag
imagemagick proot proot-distro python bsdtar mutt nmap neovim
```

Installing and Setting up Ubuntu on Termux

1.Install ubuntu

```
proot-distro install debian
```

```
proot-distro login debian
```

Inside proot-distro

```
apt update && apt upgrade
```

```
apt install apt-utils build-essential cmake neovim git wget subversion
imagemagick nano ranger
```

2.Installing python3

```
apt install python3-pip python3-numpy python3-scipy python3-matplotlib
python3-mpmath python3-sympy python3-cvxopt
```

3.Installing neovim and ranger

```
apt install neovim ranger libxtst-dev libx11-dev python3-pynvim
```

- `pip3 install ueberzug`

- Refer to this document to setup neovim and ranger.

  For usage tips refer to this document.

4.Installing LaTeX

```
apt install texlive-full gnumeric
```

Latex and Python

Latex Template

```
svn co https://github.com/gadepall/training/trunk/math

texfot pdflatex main.tex
```

Python

```
python3 codes/tri_sss.py
```

Platformio

1. Install Packages

apt update && apt upgrade

apt install apt−utils build−essential cmake neovim

apt install git wget subversion imagemagick nano

apt install avra avrdude gcc−avr avr−libc

#−−−−−−−−−−−−−−−−−End Installing ubuntu on termux

−−−−−−−−−−−−−−−−−−−−−−−−−−

#−−−−−−−−−−−−−−−−− Installing python3 on termuxubuntu

−−−−−−−−−−−−−−−−−−−−−−−−−−

apt install python3−pip python3−numpy python3−scipy python3−matplotlib python3−mpmath python3−sympy python3−cvxopt

#−−−−−−−−−−−−−−−−− End installing python3 on termuxubuntu

−−−−−−−−−−−−−−−−−−−−−−−−−−

#−−−−−−−−−−−−−−−−− Installing platformio on termuxubuntu

−−−−−−−−−−−−−−−−−−−−−−−−−−

pip3 install platformio

#−−−−−−−−−−−−−−−−− End installing python3 on termuxubuntu

−−−−−−−−−−−−−−−−−−−−−−−−−−

2. Execute the following on ubuntu

cd ide/piosetup/codes

pio run

3. Connect your arduino to the laptop/rpi and type

pio run −t nobuild −t upload

4. The LED beside pin 13 will start blinking

Arduino Droid

1. Install ArduinoDroid from apkpure
2. Open ArduinoDroid and grant all permissions
3. Connect the Arduino to your phone via USB-OTG
4. For flashing the bin files, in ArduinoDroid,

Actions−>Upload−>Upload Precompiled

then go to your working directory and select

pio/build/uno/firmwarehex

for uploading hex file to the Arduino Uno

5. The LED beside pin 13 will start blinking

Installing neovim

#Debian

sudo apt install neovim ranger libxtst-dev libx11-dev python3-pynvim

pip3 install ueberzug

#Installing vim-plug

curl -fLo ~/.local/share/nvim/site/autoload/plug.vim --create-dirs https://raw.githubusercontent.com/junegunn/vim-plug/master/plug.vim

curl -fLo ~/.config/nvim/init.vim --create-dirs https://raw.githubusercontent.com/gadepall/termux/main/neovim/init.vim

#Enter plugin

nvim ~/.config/nvim/init.vim

:PlugInstall

:qa!

#Beginners may ignore the following

#For installing a plugin
#call plug#begin('~/.config/nvim/plugged')
#Plug 'lervag/vimtex'
#call plug#end()

nvim

:UpdateRemotePlugins
:q!
#To remove a plugin
#call plug#begin('~/.config/nvim/plugged')
#Plug 'lervag/vimtex'
#call plug#end()
:w
:!source ~/.config/nvim/init.vim
:PlugClean

Enter y

#Arch
sudo pacman -Sy ranger python-pynvim ueberzug
#All other instructions remain the same

SEVEN SEGMENT

1.How to know seven segment is working or not??

Ans) sevenseg- Arduino

com pin - 5v

Dot pin – ground

check a,b,,c,d,e,f,g each pin – ground

2.How to Check seven segment Dispaly as common cathode or common anode

Ans) Common Anode : The pins need to be low to light up and common pin need to be connected to vcc.

common connected to ardunio 5v and resister one end is connected to ardunio ground and another end is connected to any one of the seven segment pin.then LED will glow.

Common Cathod:The pins need to be high to light up and common pin need to be connected to ground.

common connected to ardunio ground and resister one end is connected to ardunio 5v and another end is connected to any one of the seven segment pin.then LED will glow.

Notes: Here we are using common anode

Code:

- software to display a number
- ide/sevenseg/codes/sevenseg/sevenseg.ino

```
#include<Arduino.h>
void sevenseg(int a,int b,int c,int d,int e,int f,int g)
{
  digitalWrite(2, a);
  digitalWrite(3, b);
  digitalWrite(4, c);
  digitalWrite(5, d);
  digitalWrite(6, e);
  digitalWrite(7, f);
  digitalWrite(8, g);
}
void setup()
{
```

```
   pinMode(2, OUTPUT);
   pinMode(3, OUTPUT);
   pinMode(4, OUTPUT);
   pinMode(5, OUTPUT);
   pinMode(6, OUTPUT);
   pinMode(7, OUTPUT);
   pinMode(8, OUTPUT);
}
void loop()
{
sevenseg(1,0,0,1,1,1,1);
}
```

- *pio run*
- *upload hex file in Arduino Droid*

7474

code:

```
int Z=0,Y=0,X=1,W=1;  //int Z,Y,X,W;
int D,C,B,A;

//Code released under GNU GPL.  Free to use for anything.
void disp_7447(int D, int C, int B, int A)
{
  digitalWrite(2, A); //LSB
  digitalWrite(3, B);
  digitalWrite(4, C);
  digitalWrite(5, D); //MSB

}
// the setup function runs once when you press reset or power the board
```

```
void setup() {
   pinMode(2, OUTPUT);
   pinMode(3, OUTPUT);
   pinMode(4, OUTPUT);
   pinMode(5, OUTPUT);
   //pinMode(6,INPUT);
   //pinMode(7,INPUT);
  //pinMode(8,INPUT);
 //pinMode(9,INPUT);

}

// the loop function runs over and over again forever
void loop() {
  //W = digitalRead(6);
 //X = digitalRead(7);
//Y = digitalRead(8);
//Z = digitalRead(9);
  A=(!W&&!X&&!Y&&!Z) || (!W&&X&&!Y&&!Z) || (!W&&!X&&Y&&!Z) || (!W&&X&&Y&&!Z) ||(!W&&!X&&!Y&&Z);
  B=(W&&!X&&!Y&&!Z) || (!W&&X&&!Y&&!Z) || (W&&!X&&Y&&!Z) || (!W&&X&&Y&&!Z);
  C=(W&&X&&!Y&&!Z) || (!W&&!X&&Y&&!Z) || (W&&!X&&Y&&!Z) || (!W&&X&&Y&&!Z);
  D = (W&&X&&Y&&!Z)||(!W&&!X&&!Y&&Z);

disp_7447(D,C,B,A);
}
```

K-MAPS

without using don't cares

code:

```
//Declaring all variables as integers
```

```
int Z=0,Y=0,X=1,W=1;
int D,C,B,A;

//Code released under GNU GPL.  Free to use for anything.
void disp_7447(int D, int C, int B, int A)
{
  digitalWrite(2, A); //LSB
  digitalWrite(3, B);
  digitalWrite(4, C);
  digitalWrite(5, D); //MSB

}
// the setup function runs once when you press reset or power the board
void setup() {
    pinMode(2, OUTPUT);
    pinMode(3, OUTPUT);
    pinMode(4, OUTPUT);
    pinMode(5, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  A=(!W&&!Y || !W&&Y&&!Z);
  B=(!W&&X&&!Z || W&&!X&&!Z);
  C=(W&&X&&!Y&&!Z || !X&&Y&&!Z || !W&&Y&&!Z);
  D=(W&&X&&Y&&!Z || !W&&!X&&!Y&&Z);

disp_7447(D,C,B,A);
}
```

With using Don’t care conditions:

code:

```
//Declaring all variables as integers
int Z=0,Y=0,X=1,W=1;
int D,C,B,A;

//Code released under GNU GPL.  Free to use for anything.
void disp_7447(int D, int C, int B, int A)
{
  digitalWrite(2, A); //LSB
  digitalWrite(3, B);
  digitalWrite(4, C);
  digitalWrite(5, D); //MSB

}
// the setup function runs once when you press reset or power the board
void setup() {
    pinMode(2, OUTPUT);
    pinMode(3, OUTPUT);
    pinMode(4, OUTPUT);
    pinMode(5, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  A=(!W&&!Y || !W&&Y&&!Z);
  B=(!W&&X&&!Z || W&&!X&&!Z);
  C=(W&&X&&!Y&&!Z || !X&&Y&&!Z || !W&&Y&&!Z);
  D=(W&&X&&Y&&!Z || !W&&!X&&!Y&&Z);

disp_7447(D,C,B,A);
}
```

With don’t care:

```
//Declaring all variables as integers
int Z=0,Y=0,X=0,W=0;  // int Z,Y,X,W
int D,C,B,A;

//Code released under GNU GPL.  Free to use for anything.
void disp_7447(int D, int C, int B, int A)
{
  digitalWrite(2, A); //LSB
  digitalWrite(3, B);
  digitalWrite(4, C);
  digitalWrite(5, D); //MSB

}
// the setup function runs once when you press reset or power the board
void setup() {
    pinMode(2, OUTPUT);
    pinMode(3, OUTPUT);
    pinMode(4, OUTPUT);
    pinMode(5, OUTPUT);
    //pinMode(6,OUTPUT);
    //pinMode(7,OUTPUT);
    //pinMode(8,OUTPUT);
    //pinMode(9,OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  //W = digitalWrite(6);
  //X = digitalWrite(7);
  //Y = digitalWrite(8);
  //Z = digitalWrite(9);
  A=(!W);
  B=(!W&&X&&!Z || W&&!X);
  C=(W&&X&&!Y&&!Z || !X&&Y || !W&&Y);
  D=(W&&X&&Y || !X&&Z);

disp_7447(D,C,B,A);
}
```

7474

DECADE_COUNTER

code:

```
#include <Arduino.h>

int W,X,Y,Z;
int D,C,B,A;

void disp_7447(int D,int C,int B, int A)
{
  digitalWrite(2, A); //LSB
  digitalWrite(3, B);
  digitalWrite(4, C);
  digitalWrite(5, D); //MSB
}

void setup() {

pinMode(2, OUTPUT);
pinMode(3, OUTPUT);
pinMode(4, OUTPUT);
pinMode(5, OUTPUT);#include "Fw_global_config.h" // This defines application specific
charactersitics

#include <stdio.h>
#include "FreeRTOS.h"
#include "task.h"
#include "semphr.h"
#include "timers.h"
#include "RtosTask.h"

/* Include the generic headers required for QORC */
#include "eoss3_hal_gpio.h"
#include "eoss3_hal_rtc.h"
#include "eoss3_hal_timer.h"
#include "eoss3_hal_fpga_usbserial.h"
#include "ql_time.h"
#include "s3x_clock_hal.h"
#include "s3x_clock.h"
#include "s3x_pi.h"
#include "dbg_uart.h"
```

```
#include "cli.h"


extern const struct cli_cmd_entry my_main_menu[];


const char *SOFTWARE_VERSION_STR;


/*
 * Global variable definition
 */


extern void qf_hardwareSetup();
static void nvic_init(void);
#define GPIO_OUTPUT_MODE (1)
#define GPIO_INPUT_MODE (0)
void PyHal_GPIO_SetDir(uint8_t gpionum,uint8_t iomode);
int PyHal_GPIO_GetDir(uint8_t gpionum);
int PyHal_GPIO_Set(uint8_t gpionum, uint8_t gpioval);
int PyHal_GPIO_Get(uint8_t gpionum);


int main(void)
{
    uint32_t i=0,j=0,k=0;
    uint32_t B,C,A,D;

    SOFTWARE_VERSION_STR = "qorc-onion-apps/qf_hello-fpga-gpio-ctlr";

    qf_hardwareSetup();
    nvic_init();

    dbg_str("\n\n");
    dbg_str( "############################\n");
    dbg_str( "Quicklogic QuickFeather FPGA GPIO CONTROLLER EXAMPLE\n");
    dbg_str( "SW Version: ");
    dbg_str( SOFTWARE_VERSION_STR );
    dbg_str( "\n" );
    dbg_str( DATE " " TIME "\n" );
    dbg_str( "############################\n\n");

    dbg_str( "\n\nHello GPIO!!\n\n"); // <<<<<<<<<<<<<<<<<<<<<< Change me!

    CLI_start_task( my_main_menu );
```

```
      HAL_Delay_Init();

PyHal_GPIO_SetDir(28,0);
PyHal_GPIO_SetDir(23,0);
PyHal_GPIO_SetDir(18,0);
PyHal_GPIO_SetDir(21,1);

while(1)
{
A=PyHal_GPIO_Get(28);
B=PyHal_GPIO_Get(23);
C=PyHal_GPIO_Get(18);

D=(!A && !B && C)  (A && !B && C) || (A && B && C)

PyHal_GPIO_Set(21,D);



}
   /* Start the tasks and timer running. */
   vTaskStartScheduler();
   dbg_str("\n");

   while(1);
}

static void nvic_init(void)
 {
   // To initialize system, this interrupt should be triggered at main.
   // So, we will set its priority just before calling vTaskStartScheduler(), not the time of enabling
each irq.
   NVIC_SetPriority(Ffe0_IRQn, configLIBRARY_MAX_SYSCALL_INTERRUPT_PRIORITY);
   NVIC_SetPriority(SpiMs_IRQn,
configLIBRARY_MAX_SYSCALL_INTERRUPT_PRIORITY);
   NVIC_SetPriority(CfgDma_IRQn,
configLIBRARY_MAX_SYSCALL_INTERRUPT_PRIORITY);
   NVIC_SetPriority(Uart_IRQn, configLIBRARY_MAX_SYSCALL_INTERRUPT_PRIORITY);
   NVIC_SetPriority(FbMsg_IRQn,
configLIBRARY_MAX_SYSCALL_INTERRUPT_PRIORITY);
 }


//needed for startup_EOSS3b.s asm file
void SystemInit(void)
{
```

```
}

//gpionum --> 0 --> 31 corresponding to the IO PADs
//gpioval --> 0 or 1
#define FGPIO_DIRECTION_REG (0x40024008)
#define FGPIO_OUTPUT_REG (0x40024004)
#define FGPIO_INPUT_REG (0x40024000)
//Set GPIO(=gpionum) Mode: Input(iomode = 0) or Output(iomode = 1)
//Before Set/Get GPIO value, the direction must be correctly set
void PyHal_GPIO_SetDir(uint8_t gpionum,uint8_t iomode)
{
   uint32_t tempscratch32;

   if (gpionum > 31)
      return;

   tempscratch32 = *(uint32_t*)(FGPIO_DIRECTION_REG);
   if(iomode)
      *(uint32_t*)(FGPIO_DIRECTION_REG) = tempscratch32 | (0x1 << gpionum);
   else
      *(uint32_t*)(FGPIO_DIRECTION_REG) = tempscratch32 & (~(0x1 << gpionum));

}


//Get current GPIO(=gpionum) Mode: Input(iomode = 0) or Output(iomode = 1)
int PyHal_GPIO_GetDir(uint8_t gpionum)
{
   uint32_t tempscratch32;
   int result = 0;

   if (gpionum > 31)
      return -1;

   tempscratch32 = *(uint32_t*)(FGPIO_DIRECTION_REG);

   result = ((tempscratch32 & (0x1 << gpionum)) ? GPIO_OUTPUT_MODE :
GPIO_INPUT_MODE);

   return result;
}

//Set GPIO(=gpionum) to 0 or 1 (= gpioval)
//The direction must be set as Output for this GPIO already
//Return value = 0, success OR -1 if error
```

```
int PyHal_GPIO_Set(uint8_t gpionum, uint8_t gpioval)
{
   uint32_t tempscratch32;

   if (gpionum > 31)
      return -1;

   tempscratch32 = *(uint32_t*)(FGPIO_DIRECTION_REG);

   //Setting Direction moved out as separate API, we will only check
   //*(uint32_t*)(FGPIO_DIRECTION_REG) = tempscratch32 | (0x1 << gpionum);
   if (!(tempscratch32 & (0x1 << gpionum)))
   {
      //Direction not Set to Output
      return -1;
   }

   tempscratch32 = *(uint32_t*)(FGPIO_OUTPUT_REG);

   if(gpioval > 0)
   {
      *(uint32_t*)(FGPIO_OUTPUT_REG) = tempscratch32 | (0x1 << gpionum);
   }
   else
   {
      *(uint32_t*)(FGPIO_OUTPUT_REG) = tempscratch32 & ~(0x1 << gpionum);
   }

   return 0;
}
//Get GPIO(=gpionum): 0 or 1 returned (or in erros -1)
//The direction must be set as Input for this GPIO already
int PyHal_GPIO_Get(uint8_t gpionum)
{
   uint32_t tempscratch32;
   uint32_t gpioval_input;

   if (gpionum > 31)
      return -1;

   tempscratch32 = *(uint32_t*)(FGPIO_INPUT_REG);
   gpioval_input = (tempscratch32 >> gpionum) & 0x1;

   return ((int)gpioval_input);
}

pinMode(6, INPUT);
pinMode(7, INPUT);
pinMode(8, INPUT);
pinMode(9, INPUT);
pinMode(13, OUTPUT);
```

```
}

void loop() {

digitalWrite(13, HIGH);
delay(2000);

disp_7447(D,C,B,A);

W = digitalRead(6);
X = digitalRead(7);
Y = digitalRead(8);
Z = digitalRead(9);

A = (!W);
B = (!W&&!X&&Y) || (W&&X) || (!W&&Z);
C = (!W&&Z) || (X&&Y) || (W&&Y);
D = (!W&&!X&&!Y&&!Z)||(W&&Z);
delay(500);

W = A;
X = B;
Y = C;
Z = D;

digitalWrite(13, LOW);
delay(1000);
}
```

FSM(Finite State Machine)

DECADE COUNTER

decrementing from 9 to 0

code:

```
int Z,Y,X,W;
int D,C,B,A;
void setup(){
 pinMode(2,OUTPUT);
 pinMode(3,OUTPUT);
 pinMode(4,OUTPUT);
 pinMode(5,OUTPUT);
 pinMode(6,INPUT);
 pinMode(7,INPUT);
 pinMode(8,INPUT);
 pinMode(9,INPUT);
 pinMode(13,OUTPUT);
}
void loop(){
 digitalWrite(13,HIGH);
 delay(1000);
 D = (!Z&&!Y&&!X&&!W) || (!Z&&!Y&&!X&&W);
 C = (!Z&&!Y&&X&&!W)|| (!Z&&!Y&&X&&W) ||(!Z&&Y&&!X&&!W) || (!Z&&Y&&!X&&W);
 B = (!Z&&!Y&&X&&!W) ||(!Z&&!Y&&X&&W)|| (!Z&&Y&&X&&!W) || (!Z&&Y&&X&&W);
 A = (!Z&&!Y&&!X&&!W)|| (!Z&&!Y&&X&&!W) ||(!Z&&Y&&!X&&!W)|| (!Z&&Y&&X&&!
W)|| (Z&&!Y&&!X&&!W);
 digitalWrite(2,A);
 digitalWrite(3,B);
 digitalWrite(4,C);
 digitalWrite(5,D);
```

```
 digitalWrite(13,LOW);
 W = digitalRead(6);
 X = digitalRead(7);
 Y = digitalRead(8);
 Z = digitalRead(9);
}
```

PLATFORMIO

code:

```
#include <LiquidCrystal.h>

// Define LCD module connections
const int rs = 8;
const int en = 9;
const int d4 = 10;
const int d5 = 11;
const int d6 = 12;
const int d7 = 13;

LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

const int dataPin = 3;        // Connect to D (data) of 74HC74 IC - Arduino pin 3
const int clockPin = 4;       // Connect to CP (clock pulse) of 74HC74 IC - Arduino pin 4
const int dFlipFlopOutputPin = 5; // Connect to Q (output) of 74HC74 IC - Arduino pin 6
const int xorOutputPin = 6;  // Connect to XOR Output - Arduino pin 10

bool previousOutput = LOW;    // Variable to store the previous state of D flip-flop output (Q)
float probabilityDinTransition = 0.3;
float voltageHigh = 3.3;
float voltageLow = 0.0;
float averageVoltageX = 0.0;
unsigned long samplePeriod = 1000;

void setup() {
  pinMode(dataPin, INPUT);
```

```
  pinMode(clockPin, OUTPUT);
  pinMode(dFlipFlopOutputPin, INPUT);
  pinMode(xorOutputPin, OUTPUT);

  // Initialize the LCD
  lcd.begin(16, 2);
  lcd.clear();
  lcd.print("Average Voltage:");
  Serial.begin(9600);
}

void loop() {
  unsigned long startTime = millis(); // Get the starting time of the clock period
  // Manually set the data input (Din) to simulate transitions
   bool dataInput = digitalRead(dataPin);
  // Create a rising edge on the clock (CP) to trigger the D flip-flop
  digitalWrite(clockPin, HIGH);
  delay(1);
  digitalWrite(clockPin, LOW);
  // Read the output (Q) of the 74HC74 IC and store it in a variable
  bool dFlipFlopOutput = digitalRead(dFlipFlopOutputPin);
  // Calculate XOR Output and XOR Inverted Output
  bool xorOutput = dFlipFlopOutput ^ previousOutput;
 // bool xorInvertedOutput = !xorOutput;
  // Output the results of XOR gate to the respective pins
  digitalWrite(xorOutputPin, xorOutput);
  averageVoltageX = (averageVoltageX + (xorOutput == HIGH ? voltageHigh : voltageLow)) /
2.0;
  unsigned long elapsedTime = millis() - startTime;
  // Wait for the remaining time of the clock period before starting the next clock periodS
  if (elapsedTime < samplePeriod) {
    delay(samplePeriod - elapsedTime);
  }
//  digitalWrite(xorInvertedOutputPin, xorInvertedOutput);
  previousOutput = dFlipFlopOutput;
  // Display the average voltage at node X on the LCD
  lcd.setCursor(0, 1);
  lcd.print("        "); // Clear the previous value
  lcd.setCursor(0, 1);
  lcd.print(averageVoltageX, 2); // Display average voltage with 2 decimal places
  // Add a small delay before repeating the loop
  delay(1000);
  Serial.print("Din: ");
  Serial.print(dataInput);
  Serial.print(", D flip-flop output (Q): ");
  Serial.print(dFlipFlopOutput);
```

```
  Serial.print(", XOR Output: ");
  Serial.print(xorOutput);
  Serial.print(",averageVoltageX: ");
  Serial.println(averageVoltageX);
}
```

ASSEMBLY

counter 0 to 9

code:

```
.include"/storage/self/primary/Download/FWC/fwc-1/assembly/piosetup/m328Pdef/m328pdef.inc"
ldi r21,0b00000111
out DDRB,r21
ldi r24,0b00000100
out DDRD,r24

ldi r22,0b00000000
ldi r23,0b00000001
loop:
add r22,r23
cpi r22,0b00000101
breq check
cbi PORTD,2
cpi r22,0b00001000
brne rst
ldi r22,0b00000000
check:
mov r24,r22
out PORTD,r24
rjmp rst
rst:
mov r21,r22
out PORTB,r21
call wait

wait:
   push r16
```

```
    push r17
    push r18

    ldi r16,0x50
    ldi r17,0x00
    ldi r18,0x00

w0:
   dec r18
   brne w0 ;loop breaks after running 256 times
   dec r17
   brne w0 ;loop breaks after running 256 times
   dec r16 brne w0 ;loop breaks after running 80 times
   pop r18
   pop r17 pop r16
   rjmp loop
```

Note:1save code with .asm extension
2.include(here we have to give the m328Pdef.inc)

Process:

1.Copy the .inc file to your home directory

- cp assembly/setup/m328Pdef/m328Pdef.inc

2.go to the code file path and excute

- avra 1.asm

3.Then hex file will generate

4.open Arduino driod and upload hex file

AVR-Gcc

Execute:

1.avr−gcc/input/codes/main.c

2.make (here we are generating hex file)

3.go to arduino droid and upload hex file

code:

```
#include <avr/io.h>
#include <util/delay.h>
#include <stdlib.h>
#include <string.h>
// TYPEDEFS
typedef uint8_t byte; // changed the name


// ------------------
//LCD DRIVER ROUTINES
//
```

```
// Routines:
// LCD_Init initializes the LCD controller
// LCD_Cmd sends LCD controller command
// LCD_Char sends single ascii character to display
// LCD_Clear clears the LCD display & homes cursor
// LCD_Integer displays an integer value
// LCD_Message displays a string
// PortB is used for data communications with the HD44780-controlled LCD.
// The following defines specify which port pins connect to the controller:
#define ClearBit(x,y) x &= ~_BV(y) // equivalent to cbi(x,y)
#define SetBit(x,y) x |= _BV(y) // equivalent to sbi(x,y)
#define LCD_RS 0 // pin for LCD R/S (eg PB0)
#define LCD_E 1 // pin for LCD enable
#define DAT4 2 // pin for d4
#define DAT5 3 // pin for d5
#define DAT6 4 // pin for d6
#define DAT7 5 // pin for d7
//// The following defines are controller commands
#define CLEARDISPLAY 0x01
#define INC 0x04

void PulseEnableLine ()
{
 SetBit(PORTB,LCD_E); // take LCD enable line high
 _delay_us(40); // wait 40 microseconds
 ClearBit(PORTB,LCD_E); // take LCD enable line low
}
void SendNibble(byte data)
{
 PORTB &= 0xC3; // 1100.0011 = clear 4 data lines
 if (data & _BV(4)) SetBit(PORTB,DAT4);
 if (data & _BV(5)) SetBit(PORTB,DAT5);
 if (data & _BV(6)) SetBit(PORTB,DAT6);
 if (data & _BV(7)) SetBit(PORTB,DAT7);
 PulseEnableLine(); // clock 4 bits into controller
}
void SendByte (byte data)
{
 SendNibble(data); // send upper 4 bits
 SendNibble(data<<4); // send lower 4 bits
 ClearBit(PORTB,5); // turn off boarduino LED
}
void LCD_Cmd (byte cmd)
{
 ClearBit(PORTB,LCD_RS); // R/S line 0 = command data
 SendByte(cmd); // send it
```

```
}
void LCD_Char (byte ch)
{
 SetBit(PORTB,LCD_RS); // R/S line 1 = character data
 SendByte(ch); // send it
}
void LCD_Init()
{
 LCD_Cmd(0x33); // initialize controller
 LCD_Cmd(0x32); // set to 4-bit input mode
 LCD_Cmd(0x28); // 2 line, 5x7 matrix
 LCD_Cmd(0x0C); // turn cursor off (0x0E to enable)
 LCD_Cmd(0x06); // cursor direction = right
 LCD_Cmd(0x01); // start with clear display
 _delay_ms(3); // wait for LCD to initialize
}
void LCD_Clear() // clear the LCD display
{
 LCD_Cmd(CLEARDISPLAY);
_delay_ms(3); // wait for LCD to process command
}

void LCD_Message(const char *text) // display string on LCD
{
 //while (*text) // do until /0 character
 LCD_Char(*text++); // send char & update char pointer
}

void LCD_Integer(int data)
// displays the integer value of DATA at current LCD cursor position
{
 char st[20] = ""; // save enough space for result
 itoa(data,st,10); //
 LCD_Message(st); // display in on LCD
}

void IntegerToBinary(int num,char *binary){
	for(int i = 7; i>= 0; i--){
		binary[i] = (num % 2) + '0';
		num /= 2;
	}
	binary[8] = '\0';
	}
// MAIN PROGRAM
int main(void)
{
```

```
  float number = 27.625;
  int a = (int)number;
  float b = number - a;
  char binary_whole_part[9]="";
  char binary_fractional_part[9]="";
  IntegerToBinary(a,binary_whole_part);
  for(int i = 0; i < 8; i++){
    b *= 2;
    int bit = (int)b;
    binary_fractional_part[i] = bit + '0';
    b -= bit;
  }
  binary_fractional_part[8] = '\0';
// use PortB for LCD interface
 DDRB = 0xFF; // 1111.1111; set PB0-PB7 as outputs
 LCD_Init(); // initialize LCD controller
 LCD_Cmd(0x80);
 LCD_Char('B');
 LCD_Char('i');
 LCD_Char('n');
 LCD_Char('a');
 LCD_Char('r');
 LCD_Char('y');
 LCD_Char(':');
 LCD_Cmd(0xC0);
 for(int i = 0;i<8;i++){
      LCD_Char(binary_whole_part[i]);
 }
 LCD_Char('.');
 for(int  i = 0;i<8;i++)
 {
      LCD_Char(binary_fractional_part[i]);
 }
 while(1)
 {
}
}
```

ARM

Installations:

In Mobile

proot-distro login debian
apt update
apt upgrade

#Install ssh-server
apt install openssh-server

Install arm toolchain and hardware tools
apt install build-essential libssl-dev libffi-dev python3-dev bison flex git tcl-dev tcl-tclreadline libreadline-dev autoconf libtool make automake texinfo pkg-config libusb-1.0-0 libusb-1.0-0-dev gcc-arm-none-eabi libnewlib-arm-none-eabi telnet python3 apt-utils libxslt-dev python3-lxml python3-simplejson cmake curl python3-pip

#ARM Hello world
#Test the GNU RM GCC Embedded toolchain
arm-none-eabi-gcc --version

#Download the pygmy-sdk
cd /data/data/com.termux/files/home/
git clone --recursive https://github.com/optimuslogic/pygmy-dev

#Download the helloworld program
cd /data/data/com.termux/files/home/
mkdir arm-examples
cd arm-examples
svn co https://github.com/gadepall/vaman/trunk/arm/codes/setup/blink
cd blink/GCC_Project
make -j4

#blink.bin should be generated in ouput/bin/ directory
#Transfer .bin file to computer
scp output/bin/blink.bin pi@192.168.0.114: #(scp output/bin/blink.bin ubuntu_username@ipaddress:location(scp output/bin/bink.bin student@192.168.95.14:home/student))
Note: make sure that mobile hotspot is connected to laptop

IN LAPTOP:

#On RPI execute the following

#If Tinyfpga is not installed
git clone --recursive https://github.com/QuickLogic-Corp/TinyFPGA-Programmer-Application.git
sudo apt install python3-pip
sudo pip3 install tinyfpgab pyserial
sudo reboot

Connect the raspberry pi to Vaman through USB.
On the left of the USB port, an LED and a button can be seen. Another button is visible on the right of the USB port.

Press the right button and immediately press the left button. The green LED starts blinking. The Vaman is now in download mode.
mode.
#This will flash the .bin file to the pygmy
sudo python3 /home/pi/TinyFPGA-Programmer-Application/tinyfpga-programmer-gui.py --port /dev/ttyACM0 --m4app blink.bin --mode m4

#On termuxdebian, the source c file is located at
blink/src/main.c

#You may make changes here to modify the colour of the led etc..

#End ARM testing

Note:1.sudo python3 /home/student/TinyFPGA-Programmer-Application/tinyfpga-programmer-gui.py --port /dev/ttyACM0 --appfpga top.bin --m4app blink.bin --mode m4-fpga –reset
2. flash.sh and top.bin files are compulsory

here bin file is generated in mobile and transfer into laptop then from laptop to vaman board