RUGGED BOARD WITH LASER EMIT.

# Overview

Interfacing the laser sensor with RBA5D2X by using MRAA library and sys class.

## **HARDWARE REQUIREMENTS**

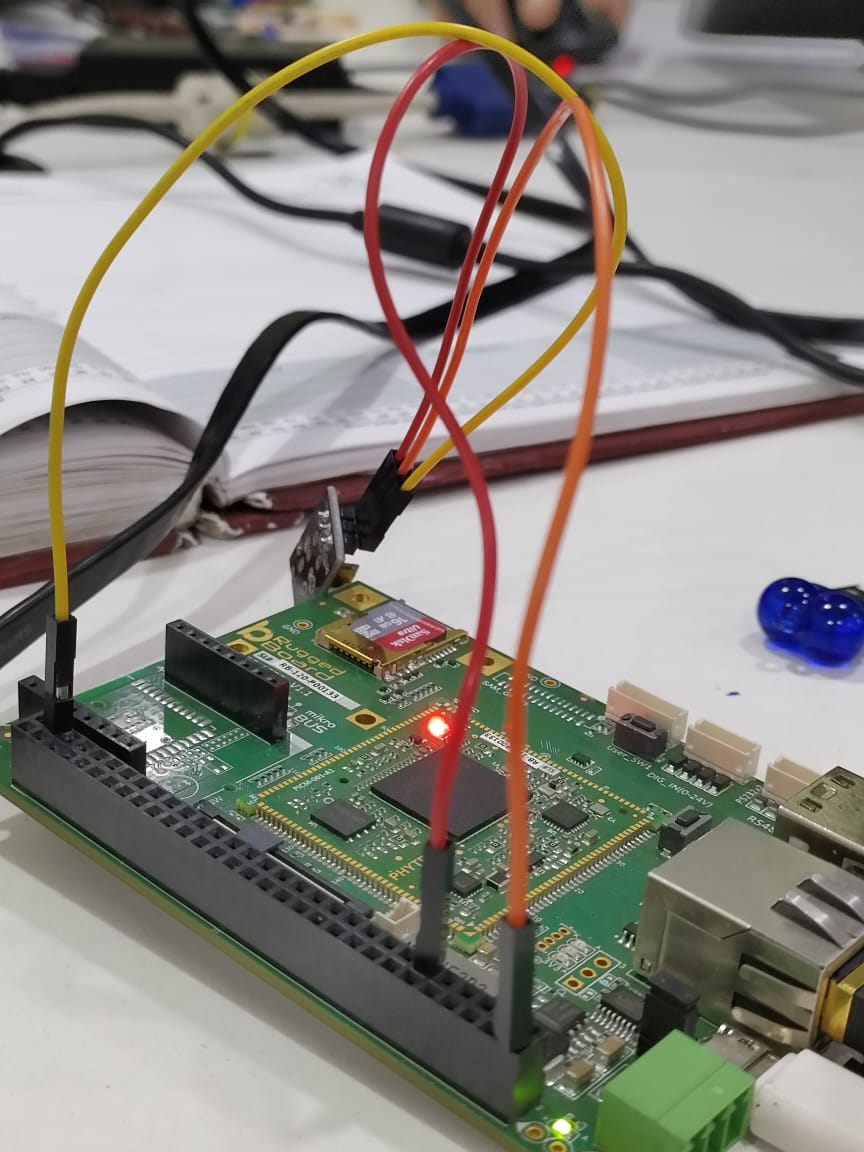
Rugged board A5D2X

KY-008 LASER EMIT MODULE

ETHERNET CABLE

SD CARD

HARDWARE CONNECTIONS



1. Connect the VCC - 5V
2. Connect the GND - GND.
3. Connect the SIgnal pin to the MRAA pin 12 in Extension Header.
4. **Red wire - signal pin , yellow - GND , Orange - 5v .**

### PROCEDURE

1.enable the tool chain by giving the below command.

. /opt/poky-tiny/2.5.2/environment-setup-cortexa5hf-neon-poky-linux-musleabi

2.write a code in host terminal and save in .c extension

3.now Make the executable file by giving belo command

${CC} <filename.c> -o <exec-name> -lmraa

4. One executable file is created . copy the file to the tftp boot directory. The command is cp <exec name > /var/lib/tftpboot/

5. Connect the rugged board and ethernet to make the connection alive.

6. Got to the root in RB and give the commands

if config eth0 <server ip>

ping<host ip >

After 0% packet loss , go to data folder

Cd /data/

7. Give the command - tftp -r <exec name> -g <give ping address>

8. ls to check if the file is available or not.

9. Give the permission for <exec file >

Chmod 777 <exec file>

10. ./<exec file>

TOGGLE THE LED BY USING USER SWITCH.

Code for MRAA

#include <mraa.h>

#include <stdint.h>

#include <stdio.h>

#include <unistd.h>

void Error\_Handler(void) {

while (1) {

// Add error handling or debugging code here

}

}

#define LED\_PIN 12 // Adjust this to the GPIO pin connected to the LED

#define USER\_SWITCH\_PIN 35 // Adjust this to the GPIO pin connected to the user switch

void GPIO\_Init(mraa\_gpio\_context \*ledPin, mraa\_gpio\_context \*userSwitchPin);

void GPIO\_Cleanup(mraa\_gpio\_context ledPin, mraa\_gpio\_context userSwitchPin);

void delay\_ms(int milliseconds);

int main(void) {

mraa\_init();

mraa\_gpio\_context ledPin, userSwitchPin;

uint8\_t ledState = 1, currentSwitchState, previousSwitchState = 1;

GPIO\_Init(&ledPin, &userSwitchPin);

while (1) {

currentSwitchState = mraa\_gpio\_read(userSwitchPin);

delay\_ms(10); // Debounce time

if (currentSwitchState != previousSwitchState && currentSwitchState == 0) {

ledState = !ledState; // Toggle the LED state

printf("LED is %s\n", ledState ? "OFF" : "ON");

mraa\_gpio\_write(ledPin, ledState);

}

previousSwitchState = currentSwitchState;

}

GPIO\_Cleanup(ledPin, userSwitchPin);

mraa\_deinit();

return 0;

}

void GPIO\_Init(mraa\_gpio\_context \*ledPin, mraa\_gpio\_context \*userSwitchPin) {

\*ledPin = mraa\_gpio\_init(LED\_PIN);

if (mraa\_gpio\_dir(\*ledPin, MRAA\_GPIO\_OUT) != MRAA\_SUCCESS) {

Error\_Handler();

}

\*userSwitchPin = mraa\_gpio\_init(USER\_SWITCH\_PIN);

if (mraa\_gpio\_dir(\*userSwitchPin, MRAA\_GPIO\_IN) != MRAA\_SUCCESS) {

Error\_Handler();

}

}

void GPIO\_Cleanup(mraa\_gpio\_context ledPin, mraa\_gpio\_context userSwitchPin) {

mraa\_gpio\_close(ledPin);

mraa\_gpio\_close(userSwitchPin);

}

void delay\_ms(int milliseconds) {

usleep(milliseconds \* 1000);

}

Code for SYS-class (without MRAA)

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#define SWITCH\_PIN "/sys/class/gpio/PC12/value" // GPIO pin for the user switch (PC12)

#define LED\_PIN "/sys/class/gpio/PD19/value" // GPIO pin for the LED (PD19)

void error(const char \*msg) {

perror(msg);

exit(1);

}

int readSwitchValue() {

FILE \*fp = fopen(SWITCH\_PIN, "r");

if (fp == NULL) {

error("Error opening switch pin");

}

char value;

fscanf(fp, " %c", &value); // Leading space to skip any leading whitespace

fclose(fp);

// '1' indicates the user switch is pressed, '0' indicates it is not pressed

return (value == '1');

}

int readLEDState() {

FILE \*fp = fopen(LED\_PIN, "r");

if (fp == NULL) {

error("Error opening LED pin");

}

char value;

fscanf(fp, " %c", &value); // Leading space to skip any leading whitespace

fclose(fp);

// '1' indicates the LED is ON, '0' indicates it is OFF

return (value == '1');

}

void toggleLED(int state) {

FILE \*ledFile = fopen(LED\_PIN, "w");

if (ledFile == NULL) {

error("Error opening LED pin");

}

// Set the LED pin based on the input state (1 for ON, 0 for OFF)

fprintf(ledFile, "%d", state);

fclose(ledFile);

}

int main() {

// Export GPIO pin for the user switch (PC12)

FILE \*exportSwitch = fopen("/sys/class/gpio/export", "w");

if (exportSwitch == NULL) {

error("Error exporting switch GPIO pin");

}

fprintf(exportSwitch, "76"); // PC12

fclose(exportSwitch);

// Set the switch pin direction to in (for reading)

FILE \*directionSwitch = fopen("/sys/class/gpio/PC12/direction", "w");

if (directionSwitch == NULL) {

error("Error setting direction for switch GPIO pin");

}

fprintf(directionSwitch, "in");

fclose(directionSwitch);

// Export GPIO pin for the LED (PD19)

FILE \*exportLED = fopen("/sys/class/gpio/export", "w");

if (exportLED == NULL) {

error("Error exporting LED GPIO pin");

}

fprintf(exportLED, "115"); // PD19

fclose(exportLED);

// Set the LED pin direction to out (for writing)

FILE \*directionLED = fopen("/sys/class/gpio/PD19/direction", "w");

if (directionLED == NULL) {

error("Error setting direction for LED GPIO pin");

}

fprintf(directionLED, "out");

fclose(directionLED);

// Read the initial state of the LED pin

int initialLEDState = readLEDState();

while (1) {

int switchState = readSwitchValue();

if (switchState) {

printf("Switch Pressed \n");

// Toggle the LED based on the initial state

toggleLED(initialLEDState ? 0 : 1);

} else {

printf("Switch Not Pressed \n");

}

// Adjust the sleep duration based on your sensor's update rate

sleep(1); // Sleep for 1 second between readings

}

// Unexport GPIO pins before exiting

FILE \*unexportSwitch = fopen("/sys/class/gpio/unexport", "w");

if (unexportSwitch == NULL) {

error("Error unexporting switch GPIO pin");

}

fprintf(unexportSwitch, "76"); // PC12

fclose(unexportSwitch);

FILE \*unexportLED = fopen("/sys/class/gpio/unexport", "w");

if (unexportLED == NULL) {

error("Error unexporting LED GPIO pin");

}

fprintf(unexportLED, "115"); // PD19

fclose(unexportLED);

return 0;

}

NOTE :

1. Procedure is same for this also but here we are Export our gpio pin and User Switch pin in rugged board terminal.
2. For export the pin give command below
3. echo <ur gpio pin no. >/sys/class/gpio/export

(refer developer.rb once if uhave doubts)

1. ls /sys/class/gpio/ - to check if the gpio is exported or not .
2. After exporting gpio pins make the tftp connection , and give permissions and execute it .
3. Choose gpio pins as per ur wish in my code i use PD19 as User switch and PC12 as gpio pin.

DONE BY ,

SHAIK MOHSEEN AHAMAD