Sistemas Embebidos de Internet de las cosas (IoT)

USANDO INTEL IoT ECLIPSE SOBRE LINUX EMBEBIDO

PONTIFICIA UNIVERSIDAD JAVERIANA

NOVIEMBRE DE 2015, BOGOTÁ COLOMBIA

INGENIERÍA ELECTRÓNICA

1. Threads

```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>

int sum; /* this data is shared by the thread(s) */
void *runner(void *param); /* the thread */
main(int argc, char *argv[])
{
  pthread_t tid1, tid2; /*the thread identifier*/
  pthread_attr_t attr; /*set of thread attributes*/
  if (argc != 2) {
    fprintf(stderr, "usage: a.out<integer value>\n");
    exit(0);
  }
  if (atoi(argv[1]) < 0) {
  fprintf(stderr, "%d must be >= 0\n",  atoi(argv[1]));
  exit(0);
  }
pthread_attr_init(&attr);
/* create the threads*/
pthread_create(&tid1,&attr,runner,argv[1]);
pthread_create(&tid2,&attr,runner,argv[1]);
pthread_join(tid1,NULL);
pthread_join(tid2,NULL);
printf("sum = %d\n",sum);
}

/*The thread will begin control in this function*/
void *runner(void *param)
{
  int upper = atoi(param);
  int i;
  sum = 0;
  if (upper > 0) {
  for (i = 1; i <= upper; i++){
  sum += i;
  printf("sum = %d, %d\n",sum,i);
  }
  }
  pthread_exit(0);
}
```

2. OnBoard LED blink (fixed frequency, without mraa)

```
#include <iostream>
#include <unistd.h>
#include <stdio.h>

using namespace std;

int main()
{
          cout << "LED Flash start" << endl;

          FILE *LEDhandler = NULL;
          char *LEDvalue = "/sys/class/gpio/gpio3/value";
```

```
        for (int i=0; i<10; i++){
                if( (LEDhandler = fopen(LEDvalue, "r+")) != NULL ){
                        fwrite("1", sizeof(char), 1, LEDhandler);
                        fclose(LEDhandler);
                }
                sleep(3);
                if( (LEDhandler = fopen(LEDvalue, "r+")) != NULL ){
                        fwrite("0", sizeof(char), 1, LEDhandler);
                        fclose(LEDhandler);
                }
                sleep(3);
        }

        cout << "LED Flash end" << endl;
        return 0;
}
```

3. Control a LED (PWM) depending on the light intensity

–Activate a digital output if the luminosity of the room is below a certain level.

```
#include "mraa.hpp"
#include <iostream>
#include <unistd.h>
#include <signal.h>


int running = 0;


int main()
{
        mraa_platform_t platform = MRAA_INTEL_GALILEO_GEN1;

        mraa::Gpio* d_pin = NULL;

        d_pin = new mraa::Gpio(32, true, true);
        d_pin->dir(mraa::DIR_OUT) ;

        mraa::Aio* a_pin = new mraa::Aio(0);
        uint16_t adc_value;
        float adc_value_float;

        if (a_pin == NULL) {
                std::cerr << "Can't create mraa::Aio object, exiting" << std::endl;
                return MRAA_ERROR_UNSPECIFIED;
        }

        // loop forever printing the input value every second
        for (;;) {
                adc_value_float = a_pin->readFloat();
                std::cout << "analog input value " << adc_value_float << std::endl;
                if(adc_value_float<0.1){
                        d_pin->write(1);
                }else{
                        d_pin->write(0);
                }
        }

        return MRAA_SUCCESS;
}
```

-**Extend the functionality to continuously (PWM) control an LED to maintain a constant luminosity.**

```
#include "mraa.hpp"
#include <iostream>
#include <unistd.h>
#include <signal.h>

int running = 0;

void
sig_handler(int signo)
{
    if (signo == SIGINT) {
        printf("closing PWM nicely\n");
        running = -1;
    }
}

int main()
{
          mraa_platform_t platform = MRAA_INTEL_GALILEO_GEN1;
          signal(SIGINT, sig_handler);

           mraa::Pwm* pwm;
          pwm = new mraa::Pwm(3);
    if (pwm == NULL) {
        return MRAA_ERROR_UNSPECIFIED;
    }
    pwm->enable(true);


    fprintf(stdout, "Cycling PWM on IO3 (pwm3) \n");
    pwm->enable(true);
          mraa::Aio* a_pin = new mraa::Aio(0);
          if (a_pin == NULL) {
                    std::cerr << "Can't create mraa::Aio object, exiting" << std::endl;
                    return MRAA_ERROR_UNSPECIFIED;
          }

          // loop forever printing the input value every second
          for (;;) {

               uint16_t adc_value;
         float adc_value_float;
                    adc_value_float = a_pin->readFloat();
                    std::cout << "analog input value " << adc_value_float << std::endl;
                    pwm->write(0.8-adc_value_float);
          }

          return MRAA_SUCCESS;
}
```

4. Sockets

Server.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

void error(const char *msg)
```

```
{
    perror(msg);
    exit(1);
}

int main(int argc, char *argv[])
{
     int sockfd, newsockfd, portno;
     socklen_t clilen;
     char buffer[256];
     struct sockaddr_in serv_addr, cli_addr;
     int n;
     if (argc < 2) {
         fprintf(stderr,"ERROR, no port provided\n");
         exit(1);
     }
     sockfd = socket(AF_INET, SOCK_STREAM, 0);
     if (sockfd < 0)
        error("ERROR opening socket");
     bzero((char *) &serv_addr, sizeof(serv_addr));
     portno = atoi(argv[1]);
     serv_addr.sin_family = AF_INET;
     serv_addr.sin_addr.s_addr = INADDR_ANY;
     serv_addr.sin_port = htons(portno);
     if (bind(sockfd, (struct sockaddr *) &serv_addr,
              sizeof(serv_addr)) < 0)
              error("ERROR on binding");
     listen(sockfd,5);
     clilen = sizeof(cli_addr);
     newsockfd = accept(sockfd,
                 (struct sockaddr *) &cli_addr,
                 &clilen);
     if (newsockfd < 0)
          error("ERROR on accept");

    for (;;) {

     bzero(buffer,256);
     n = read(newsockfd,buffer,255);
     if (n < 0) error("ERROR reading from socket");
     printf("Here is the message: %s\n",(int)buffer);
     n = write(newsockfd,"Llego el dato",13);
     if (n < 0) error("ERROR writing to socket");
    }
     close(newsockfd);
     close(sockfd);
     return 0;
}
```

Client.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include "mraa/aio.h"
#include <math.h>

void error(const char *msg)
{
    perror(msg);
    exit(0);
}

int main(int argc, char *argv[])
{
```

```
        mraa_aio_context adc_a0;
        uint16_t adc_value = 0;
        adc_a0 = mraa_aio_init(0);

  if (adc_a0 == NULL) {
       return 1;
         }
  float resistance=0;
  float temperature;

  int B = 4275;

  int sockfd, portno, n;
  struct sockaddr_in serv_addr;
  struct hostent *server;

  char buffer[256];
  if (argc < 3) {
     fprintf(stderr,"usage %s hostname port\n", argv[0]);
     exit(0);
  }
  portno = atoi(argv[2]);
  sockfd = socket(AF_INET, SOCK_STREAM, 0);
  if (sockfd < 0)
      error("ERROR opening socket");
  server = gethostbyname(argv[1]);
  if (server == NULL) {
      fprintf(stderr,"ERROR, no such host\n");
      exit(0);
  }
  bzero((char *) &serv_addr, sizeof(serv_addr));
  serv_addr.sin_family = AF_INET;
  bcopy((char *)server->h_addr,
       (char *)&serv_addr.sin_addr.s_addr,
       server->h_length);
  serv_addr.sin_port = htons(portno);
  if (connect(sockfd,(struct sockaddr *) &serv_addr,sizeof(serv_addr)) < 0)
      error("ERROR connecting");

  char *mensaje = "La temperatura es: ";
  for (;;) {
    adc_value = mraa_aio_read(adc_a0);
    resistance = (float)(((1023-adc_value)*10000)/adc_value);
    temperature=1/(log(resistance/10000)/B+1/298.15)-273.15;
    snprintf(buffer, sizeof buffer, "%s%f", mensaje, temperature);

    n = write(sockfd,buffer,strlen(buffer));
    if (n < 0)
         error("ERROR writing to socket");
    bzero(buffer,256);
    n = read(sockfd,buffer,255);
    if (n < 0)
         error("ERROR reading from socket");
    printf("%s\n",buffer);
          sleep(1);
  }

  close(sockfd);
  mraa_aio_close(adc_a0);
         return MRAA_SUCCESS;
}
```

5. Simple Webserver

```
<html>
<head>
<title> Mi primera pagina WEB </title>
</head>
<body>
<h1> Mi primera pagina</h1>
<p> test en la Galileo </p>
<img src=foto.jpg></img>
</body>
</html>
```

6. Advanced Webserver

```
#include "mraa/aio.h"
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <math.h>

int main()
{
          mraa_aio_context adc_a0;
          uint16_t adc_value = 0;
          adc_a0 = mraa_aio_init(0);

           if (adc_a0 == NULL) {
                   return 1;
           }
           char *comandobase = "iotkit-admin observation temperature ";
           char comando[256];
           float resistance=0;
           float temperature;
           int B = 4275;

  for (;;) {
            adc_value = mraa_aio_read(adc_a0);
            resistance = (float)(((1023-adc_value)*10000)/adc_value);
            temperature=1/(log(resistance/10000)/B+1/298.15)-273.15;
            fprintf(stdout, "La temperatura es:  %.5f\n", temperature);

            snprintf(comando, sizeof comando, "%s%f", comandobase, temperature);
            system(comando);
      sleep(10);
  }

    mraa_aio_close(adc_a0);
          return MRAA_SUCCESS;
}
```