FENCING SCORING MACHINE

DECEMBER 28, 2019

AIN SHAMS UNIVERSITY – FACULTY OF ENGINEERING

FACULTY OF ENGINEERING

AIN SHAMS UNIVERSITY

CSE 316: Microntrollers and Interfacing

Fencing Scoring Machine

Submitted by:

Abdelrahman Amr Issawi	aid-issawi@hotmail.com	16P6001@eng.asu.edu.eg
Abdelrahman Ibrahim ELGhamry	Ghamry98@hotmail.com	16P3043@eng.asu.edu.eg
Hossam ELDin Khaled	hossampen97@gmail.com	16P3025@eng.asu.edu.eg
Mayar Khaled Mohamed	Mayarkhaledd_3@yahoo.com	16P6030@eng.asu.edu.eg

Submitted to:

Dr. Haytham Azmy

DECEMBER 28, 2019

A REPORT FOR MICROCONTROLLERS AND INTERFACING COURSE CODDED CSE316 WITH THE REQUIREMENTS OF AIN SHAMS UNIVERSITY

1. INTRODUCTION

Watching a fencing match can be confusing. For a new spectator, it's often just a flurry of swords, flashing lights, and the director making arcane hand signals. This can make it extremely difficult to get your footing and actually understand what is going on during a match.

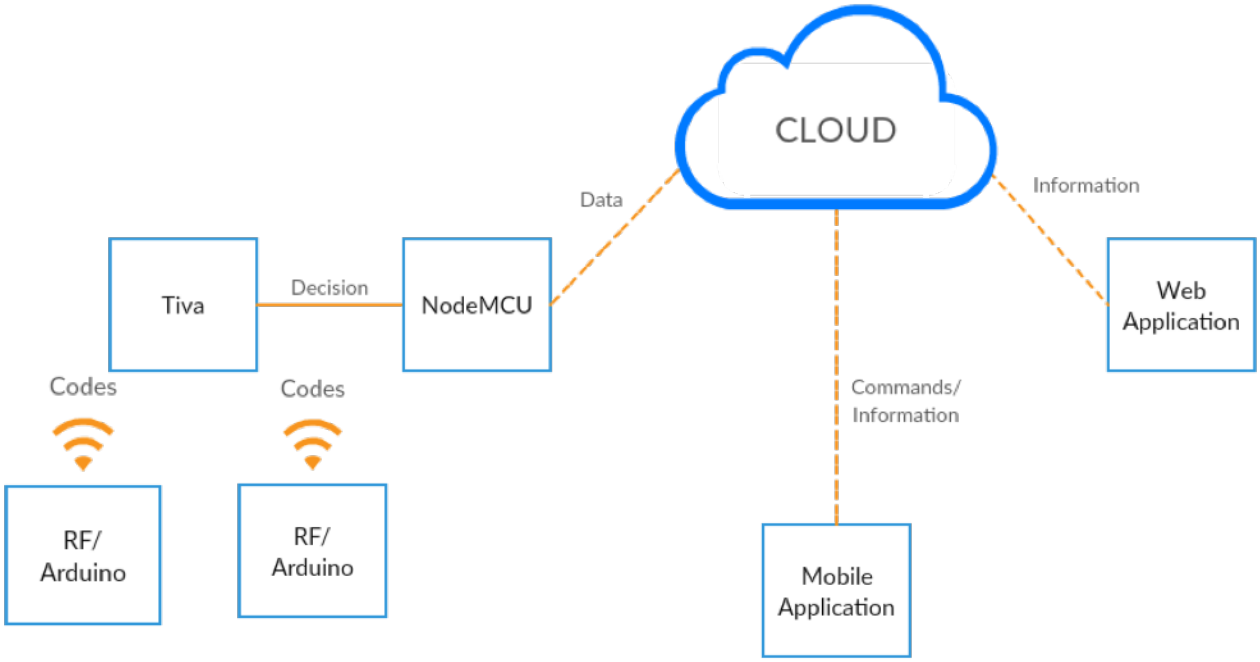
The match starts and the 2 opponents aim to score a point, to score a point a fencer must hit their opponent with their sword's end before they get hit. The sword's end can land anywhere on the opponent's body, from their head to their toes.

The match ends whenever a fencer reaches 15 point, but the question is, how the points are calculated? how are they incremented?

Traditionally, Each fencer has a wire attached to it from an end and the other end is attached to a traditional scoring machine when ever a fencer hits the other one a buzzer works, colored (red or green) light on that fencer's side of the machine lights up, and the referee scores the touch manually on paper.

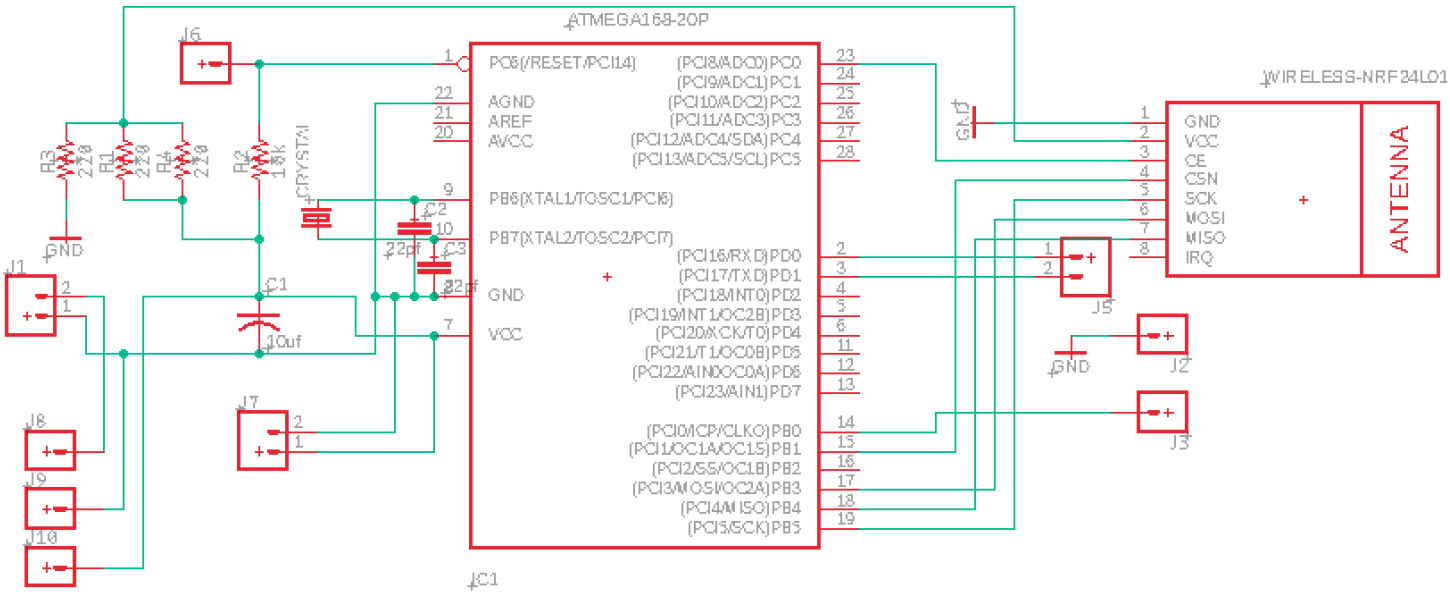
In this project we are upgrading this system, for a better performance and less cost. The idea is the same, whenever a fencer hits the other his score increase, but instead of wired system we are considering using wireless connection between the fencer and the machine, in addition we implemented a web application to display the score and automatically increment the valid points or decrement the invalid ones, Finally the web application can be managed and controlled using a mobile application.

2. BLOCK DIAGRAM

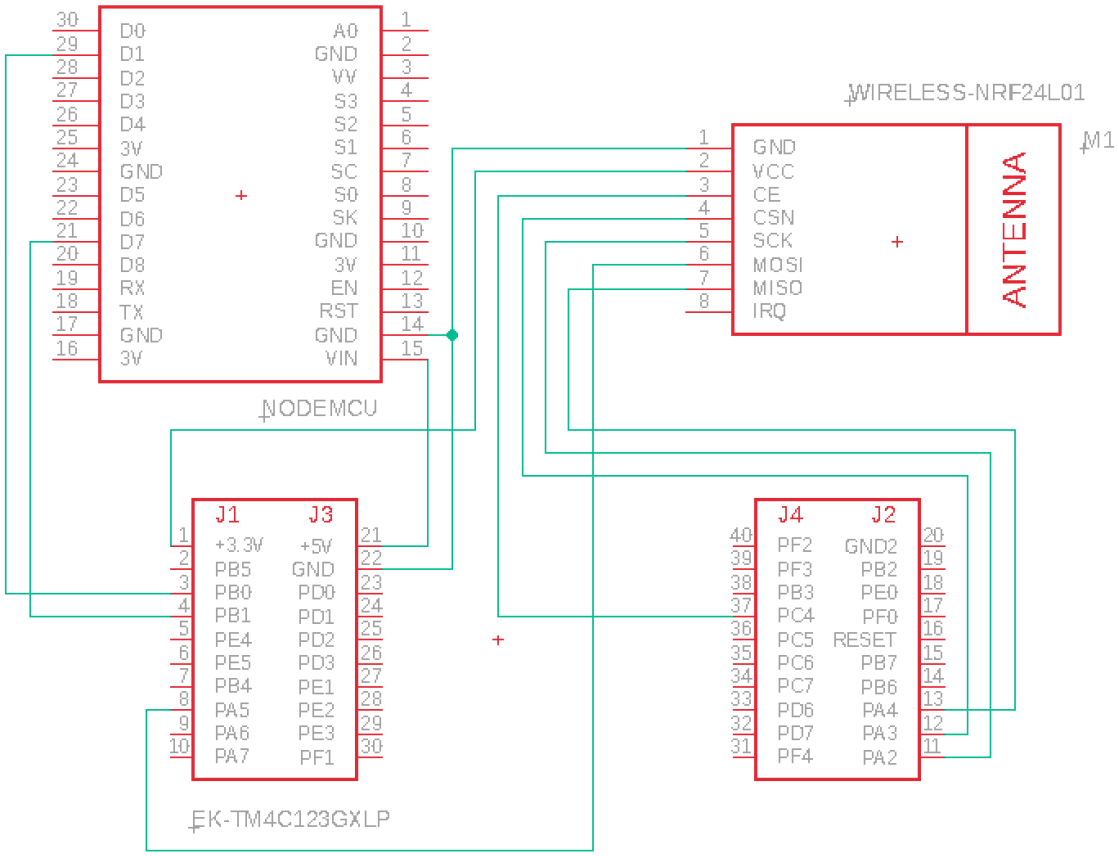


CIRCUIT DIAGRAM

RF/Arduino



Tiva/Nodemcu/RF



3. COPY OF THE CODE

Tiva code:

```
#include <stdint.h>
#include <stdbool.h>

#include "inc/hw_memmap.h"
#include "inc/hw_ints.h"

#include "driverlib/fpu.h"
#include "driverlib/gpio.h"
#include "driverlib/interrupt.h"
#include "driverlib/pin_map.h"
#include "driverlib/sysctl.h"

#include "nrf24l01.h"

#include "tm4c123gh6pm.h"
#include "delay.h"
#include "reg.h"


struct nrf24l01p nrf;
uint8_t addresses[][6] = {"1Node","2Node"};

#define player01        1u
#define player02        1u << 1

void running(){
    while(1) {
        if (nrf24l01p_available(&nrf)) {
            uint8_t payload;

            //get the payload
            while (nrf24l01p_available(&nrf)) {
                nrf24l01p_read(&nrf, &payload,
                    sizeof(payload));
            }

            if(payload == 0x37){
                GPIO_PORTB_DATA_R &= ~(0x01);
                delayTimer(1);
                GPIO_PORTB_DATA_R |= 0x01;
                delayTimer(1);
            }
            else if(payload == 0x27){
                GPIO_PORTB_DATA_R &= ~(0x02);
                delayTimer(1);
                GPIO_PORTB_DATA_R |= 0x02;
```

```
                delayTimer(1);
            }

        }

        //give it a little space
        SysCtlDelay((SysCtlClockGet() >> 12) * 5);
    }
}

int main(void) {
     //
     // Enable lazy stacking for interrupt handlers.  This
     allows floating-point
     // instructions to be used within interrupt handlers, but
     at the expense of
     // extra stack usage.
     //
     FPULazyStackingEnable();
     FPUEnable();

     //
     // Set the clocking to run directly from the crystal.
     // Clock to 80MHZ
     //
     SysCtlClockSet(SYSCTL_SYSDIV_2_5 | SYSCTL_USE_PLL |
     SYSCTL_XTAL_16MHZ | SYSCTL_OSC_MAIN);
     IntMasterEnable();

     //set up the radio
     if( !nrf24l01p_setup(&nrf, GPIO_PORTC_BASE, GPIO_PIN_4,
     SSI0_BASE) ){
          //we couldn't communicate with the radio
          return 1;
     }

     nrf24l01p_set_PA_level(&nrf, RF24_PA_MAX);

     nrf24l01p_open_reading_pipe(&nrf, 1, addresses[0]);
     nrf24l01p_open_reading_pipe(&nrf, 2, addresses[1]);

     nrf24l01p_start_listening(&nrf);

     setPortBGPIO(player01, player02);

     running();

     return 0;
     }
```

NodeMCU code:

```
#include <Arduino.h>

#include <ESP8266WiFi.h>
#include <ESP8266WiFiMulti.h>

#include <ESP8266HTTPClient.h>

#include <WiFiClient.h>

#include <BlynkSimpleEsp8266.h>

#include <SoftwareSerial.h>

ESP8266WiFiMulti WiFiMulti;

WiFiClient client;
HTTPClient http;

//rx tx
//SoftwareSerial tiva(D6,D5);

void ICACHE_RAM_ATTR readTiva();
void ICACHE_RAM_ATTR readTiva2();

int redScore = 0;
int blueScore = 0;
boolean isMatchRunning = false;

boolean newPoint = 0;

long last_t;

void setup() {

 Serial.begin(115200);

 pinMode(5, INPUT_PULLUP);
 attachInterrupt(digitalPinToInterrupt(5), readTiva2, RISING);

 pinMode(13, INPUT_PULLUP);
 attachInterrupt(digitalPinToInterrupt(13), readTiva, FALLING);

 Blynk.begin("Twl7IpLaed9RZPZkiSJUEi-hM03Ju1bg", "Mayar", "p1234567", "blynk-
 cloud.com");

 WiFi.mode(WIFI_STA);
```

```
  WiFiMulti.addAP("Mayar", "p1234567");

}

void setValue(String pin, int value){
  // wait for WiFi connection
  if ((WiFiMulti.run() == WL_CONNECTED)) {
    if (http.begin(client, "http://blynk-cloud.com/Twl7IpLaed9RZPZkiSJUEi-
          hM03Ju1bg/update/" + pin + "?value=" + value)) {  // HTTP
        // start connection and send HTTP header
        int httpCode = http.GET();

        // httpCode will be negative on error
        if (httpCode > 0) {
          if (httpCode == HTTP_CODE_OK || httpCode == HTTP_CODE_MOVED_PERMANENTLY)
              {
            String payload = http.getString();
            //Serial.println(payload);
            newPoint = false;
          }
      }
    }
  }
}

void getValue(String pin) {
  // wait for WiFi connection
  if ((WiFiMulti.run() == WL_CONNECTED)) {
    if (http.begin(client, "http://blynk-cloud.com/Twl7IpLaed9RZPZkiSJUEi-hM03Ju1bg/get/"
          + pin)) {  // HTTP
        // start connection and send HTTP header
        int httpCode = http.GET();

        // httpCode will be negative on error
        if (httpCode > 0) {
          if (httpCode == HTTP_CODE_OK || httpCode == HTTP_CODE_MOVED_PERMANENTLY)
              {
            String payload = http.getString();
            int val = 0;
            if(payload[3] == '"'){
               val = payload[2] - 48;
            }
            Else {
              val = (payload[2] - 48)*10 + (payload[3] - 48);
            }
            if(pin == "v0"){
              redScore = val;
```

```
          }
          else if(pin == "v1"){
            blueScore = val;
          }
          else if(pin == "v2"){
            isMatchRunning = val;
          }
        }
      }
    }
  }
}

void readTiva(){
  long nt = millis();
  Serial.println("ISR");
  if(nt - last_t > 1000){
    Serial.println("inside");
    if(isMatchRunning){
      redScore++;
      Serial.println(redScore);
      newPoint= true;
    }
  }
  last_t = millis();
}
void readTiva2(){
  long nt = millis();
  Serial.println("ISR2");
  if(nt - last_t > 1000){
    Serial.println("inside");
    if(isMatchRunning){
      blueScore++;
      Serial.println(blueScore);
      newPoint = true;
    }
  }
  last_t = millis();
}

void loop() {
  getValue("v2");
  if(!newPoint){
     getValue("v0");
     getValue("v1");
  }
  if(isMatchRunning && newPoint) {
```

```
   setValue("v0", redScore);
   setValue("v1", blueScore);
  }
  Blynk.run();

  delay(1000);
}
```

Arduino/RF code:

```
#include <SPI.h>
#include "RF24.h"

/*  Hardware configuration: Set up nRF24L01 radio on SPI bus plus pins A0 & D9
 *  CE, CSN
 */
RF24 radio(A0, 9);

byte addresses[][6] = {"1Node","2Node"};

int analogValue = 0;

void setup() {
  Serial.begin(115200);

  radio.begin();
  radio.setPALevel(RF24_PA_MAX);

  radio.openWritingPipe(addresses[0]);
  //radio.openWritingPipe(addresses[1]);

  radio.setDataRate(RF24_1MBPS);
  //radio.setCRCLength(RF24_CRC_16) ;
  radio.setChannel(76);
  radio.stopListening();

  pinMode(8, INPUT_PULLUP);
}

boolean sendCode(int code){
  // First, stop listening so we can talk.

  unsigned long started_waiting_at = micros();

  boolean sent = radio.write( &code, sizeof(code) );

  // Set up a variable to indicate if a response was received or not
```

```
  boolean timeout = false;

  while ( ! radio.available() ){                          // While nothing is received
    if (micros() - started_waiting_at > 200000 ){         // If waited longer than 200ms, indicate
timeout and exit while loop
        timeout = true;
        break;
    }
  }

  return !timeout | sent;
}

void loop() {
  if(!digitalRead(8)) {
    Serial.println("Im pressed");
    if(!sendCode(0x27)){
    //if(!sendCode(0x37)){
      Serial.println("Sending Failed!");
    }
    delay(100);
  }
}
```

4. MAIN CHALLENGES/ LIMITATIONS

- Finding cost effective wireless modules.
- Understanding and implementing SPI protocol that the wireless modules depend on.
- Data synchronization and locking mechanisms to achieve data integrity between the different nodes in the system that read and update the same variables.
  System nodes:
  - Web server.
  - Mobile application.
  - Tiva c micro controller with nodemcu as a wifi module.

  Shared variables:
  - Score for each player.
  - State of match running or not.

5. VIDEO LINK

https://drive.google.com/file/d/15YP-DJWKdDIflwZZFgw4NMKGQudATXu5/view?usp=sharing