Victor-2567375 Marco-2554643 Jannis-2555689

A1)

```
#include <SPI.h>
#include <Ethernet.h>
// Einfacher Webserver
// Im Browser http://<IP-Adresse>/ eingeben
//  z.B.http://192.168.0.42/
// Einrichtung: siehe Echo-Programm
int pin_motion = 27;
int pin_temperature = 51;
int t = 0;
int t_tmp = 0;
unsigned long bewegungen[10] = {0};
int item = 0;
// MAC-Adresse
byte mac[] = {
  0xAF, 0xFE, 0xAB, 0xBA, 0xDE, 0xAF
};

// IP-Adresse
IPAddress ip(192, 168, 0, 42);

// Ob DHCP benutzt werden soll
const bool USE_DHCP =true;

// Server
EthernetServer server(80);// HTTP-Protokoll

//LISTE
typedef struct queue_struct{
  int value;
  struct queue_struct* next;
};

struct queue_struct * head ;// HEAD-Element, dass stets auf das erste Element
void addElement_head(struct queue_struct* newElem){
  // Fuege Element an Anfang der Liste ein
  newElem->next = head;
  head = newElem;
}

struct queue_struct * newElem(int value ) {
// erzeuge neues Element und caste void - Pointer aufrichtigen Typ
  struct queue_struct * newQueue = ( struct queue_struct* ) malloc(sizeof(stru
  newQueue->value = value ;
  // default pointers
  newQueue->next = NULL ;
  return new struct queue_struct;
```

```
}

unsigned long delta_t(){
  unsigned long ret  =millis()-t;
  return ret;
}

void setup() {
  // Serielle Schnittstelle zur Ausgabe der IP nutzen
  Serial.begin(9600);

  // Ethernet-Verbindung und Server starten
  Serial.println("IP-Adresse ueber DHCP anfordern...");
  if (!USE_DHCP ||Ethernet.begin(mac) == 0)
  {
      Serial.println("DHCP-Anfrage fehlgeschlagen - manuelle Konfiguration");
      Ethernet.begin(mac, ip);
  }
  server.begin();
  Serial.print("Arduino hat die IP-Adresse ");
  Serial.println(Ethernet.localIP());

  pinMode(pin_motion, INPUT);
  pinMode(pin_temperature, INPUT);
}

// Puffer fuer das empfangene Kommando vom Browser
const int COMMAND_SIZE = 2048;
char command[COMMAND_SIZE];
EthernetClient client;

bool had_movement =false;

void do_measurements() {
  // TODO: Hier messen
  if(digitalRead(pin_motion) == HIGH){
     had_movement =true;
     t =millis();
    }
    if(delta_t()>1800000){
       had_movement =false;
    }
  if(!had_movement || delta_t()>1800000){
    bewegungen[item%10] =millis();
    item++;
     t =millis();
     had_movement =true;
  }
  if(millis()%3600000 == 0){
```

```
    unsigned long now = delta_t();
    struct queue_struct * n  = (struct queue_struct *) malloc(sizeof(struct qu
    int reading =analogRead(pin_temperature);
    n->value=map(reading, 0, 410, -50, 150);
     addElement_head(n);
      }
}


void display_page() {
  client.println("<h1>Mein Arduino</h1>");
  // TODO: Hier HTML-Seite ausgeben
  //keine zeit mehr >.<
}



void loop() {
  do_measurements();

  // Anfrage bearbeiten
  client = server.available();
  if (client) {
    int currentLineIsBlank = 1;
    int cmd = 0;
    while (client.connected()) {  // Client verbunden
      if (client.available()) {   // Zeichen verfuegbar
        char c = client.read();
        Serial.write(c);
        if (cmd < COMMAND_SIZE - 1)
            command[cmd++] = c;
        if (c  =='\n' && currentLineIsBlank) {
           command[cmd] ='\0';

          // Antwort schicken
          client.println("HTTP/1.1 200 OK");
          client.println("Content-Type: text/html");
          client.println("Connection: close");
          client.println("Refresh: 5");
          client.println();

          client.println("<!DOCTYPE HTML>");
          client.println("<html><head><title>MeinArduino</title></head><body>

           display_page();

           client.println("</body></html>");
           break;
        }
```

```
        if (c =='\n') {
          currentLineIsBlank = 1;
        }
        else if (c !='\r') {
          currentLineIsBlank = 0;
        }
      }
    }
    delay(1);
    client.stop();
  }
}
```

A2)

```
#include <SPI.h>
#include <Ethernet.h>
#include <LiquidCrystal.h>

// Einfacher Webserver
// Im Browser http://<IP-Adresse>/ eingeben
//  z.B.http://192.168.0.42/

// Einrichtung: siehe Echo-Programm

// MAC-Adresse
byte mac[] = {
  0xAF, 0xFE, 0xAB, 0xBA, 0xDE, 0xAF
};

// IP-Adresse
IPAddress ip(192, 168, 0, 42);

// Ob DHCP benutzt werden soll
const bool USE_DHCP =true;

// Server
EthernetServer server(80);// HTTP-Protokoll

// Display
unsigned const PIN_BACKLIGHT = 51;
LiquidCrystal lcd(8, 9, 2, 3, 4, 5);// set up LCD

void lcd_backlight(bool light) {
  digitalWrite(PIN_BACKLIGHT, light);
}

void setup() {
  // Serielle Schnittstelle zur Ausgabe der IP nutzen
```

```
  Serial.begin(9600);

  // Ethernet-Verbindung und Server starten
  Serial.println("IP-Adresse ueber DHCP anfordern...");
  if (!USE_DHCP ||Ethernet.begin(mac) == 0)
  {
      Serial.println("DHCP-Anfrage fehlgeschlagen - manuelle Konfiguration");
      Ethernet.begin(mac, ip);
  }
  server.begin();
  Serial.print("Arduino hat die IP-Adresse ");
  Serial.println(Ethernet.localIP());

  lcd.begin(16, 2); // init display with 16 columns and 2 rows
  lcd.setCursor(0,0); // setCursor(column, row);

  pinMode(PIN_BACKLIGHT, OUTPUT);
  lcd_backlight(true);
}

// Puffer fuer das empfangene Kommando vom Browser
const int COMMAND_SIZE = 2048;
char command[COMMAND_SIZE];
EthernetClient client;


//parse zwei chars ein, füge sie zu einer hexZahl zusammen
//und gib den char zurück
char parseToChar(char  first,char second){
  int erg = 0;
  int a  =int(first)>=65?int(first)-55:int(first)-48;
  int b  =int(second)>=65?int(second)-55:int(second)-48;
  erg = a*10+b;
  return char(erg);
}

void handle_get(const char path[]) {
  Serial.print("GET: ");
  Serial.println(path);
  // TODO: LCD-Backlight an/ausschalten
  // TODO: path auf LCD ausgeben
  if  (strncmp(path,"/on", 3) == 0) {
    lcd_backlight(true);
   }else if  (strncmp(path,"/off", 4) == 0) {
            lcd_backlight(false);
  }else{
      lcd.clear();
      int i = 1;
      while(path[i] != '\0'){
```

```
        if(path[i] == '+'){
          lcd.print(' ');
          i++;
          continue;
        }
        if(path[i] == '%'){
          char help = parseToChar(path[i+1], path[i+2]);
          lcd.print(help);
          i = i+2;
          continue;
            }
        lcd.print(path[i]);
        if(i%16 == 0){
          lcd.println();
        }
        i++;
         }
  }
}


void loop() {
  // Anfrage bearbeiten
  client = server.available();
  if (client) {
    int currentLineIsBlank = 1;
    int cmd = 0;
    while (client.connected()) {  // Client verbunden
      if (client.available()) {   // Zeichen verfuegbar
        char c = client.read();
        Serial.write(c);
        if (cmd < COMMAND_SIZE - 1)
            command[cmd++] = c;
        if (c  =='\n' && currentLineIsBlank) {
           command[cmd] ='\0';

          // zerlegen des strings
          char *method = strtok(command," ");
          char *path = strtok(NULL," ");

          if  (strcmp(method,"GET") == 0) {
           handle_get(path);
          }

          // Antwort schicken
          client.println("HTTP/1.1 200 OK");
          client.println("Content-Type: text/html");
          client.println("Connection: close");
          client.println();
```

```
          client.println("<!DOCTYPE HTML>");
          client.println("<html><head><title>MeinArduino</title></head><body>

           client.println("</body></html>");
           break;
         }
         if (c =='\n') {
          currentLineIsBlank = 1;
         }
         else if (c !='\r') {
          currentLineIsBlank = 0;
         }
       }
     }
     delay(1);
     client.stop();
   }
}
```