Protokol o semestrálním projektu z předmětu Elektronika a komunikace 2024

Název projektu: Flight controller na quad

Autor: Matej Bryja

Popis zapojení:
Na zachovanie kompaktných rozmerov som sa rozhodol použiť iba modul s ESP32. To však znamená, že je potrebné riešiť viaceré veci navyše. Napríklad na programovanie je nevyhnutný USB na UART prevodník. Mikrokontrolér komunikuje s perifériami prostredníctvom rôznych protokolov.

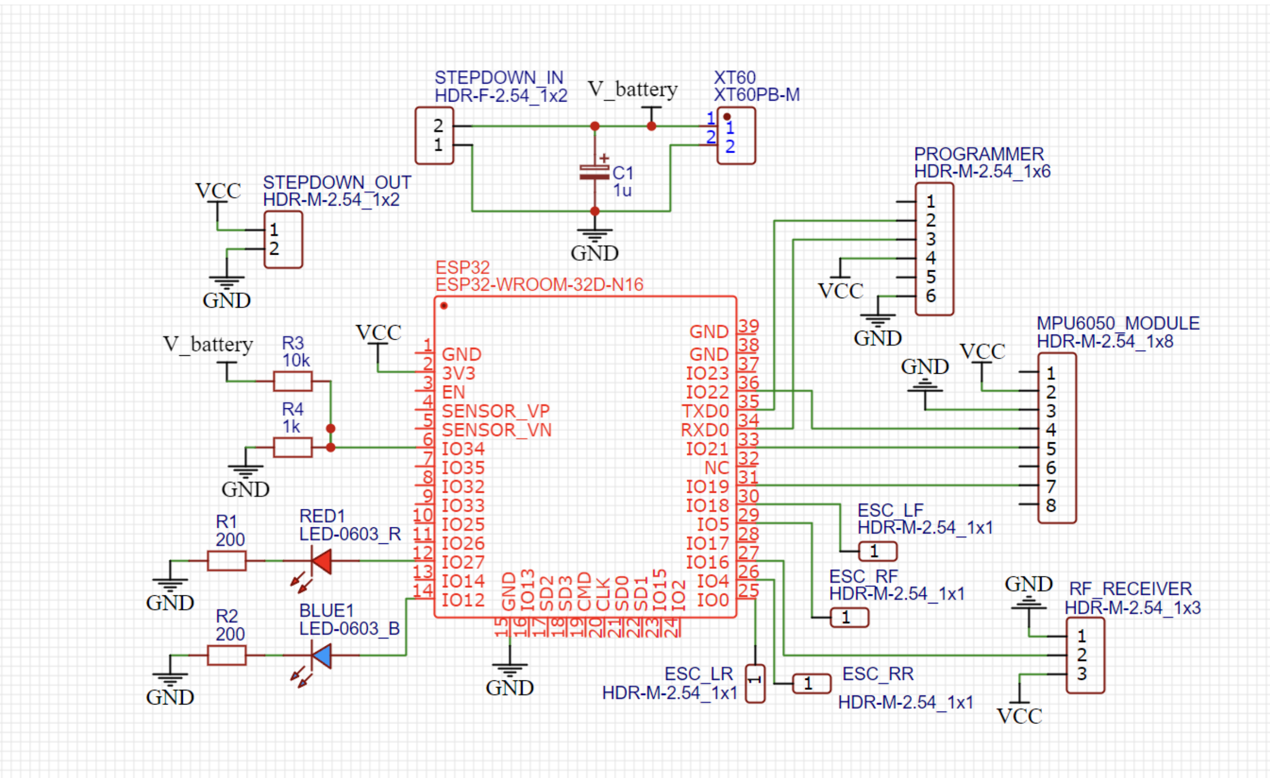
Hlavným senzorom je inerciálna jednotka **MPU6050**, ktorá meria uhlovú rýchlosť a uhol náklonu dronu. Na ovládanie používam rádiový vysielač a prijímač **FLY-SKY FS-I6X**. Vrtule sú poháňané štyrmi bezkartáčovými motormi, riadenými elektronickými regulátormi otáčok (**ESC**).

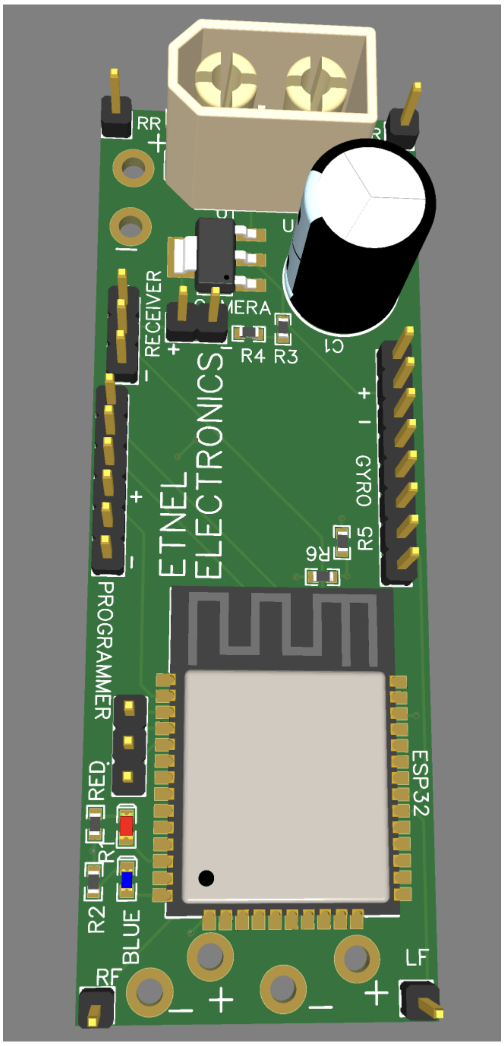
Napájanie zabezpečuje 4S Li-Po batéria s kapacitou 1550 mAh v kombinácii s DC-DC meničom, ktorý napája riadiacu elektroniku. Na priebežné meranie napätia batérie využívam odporový delič a indikačné LED diódy.

Pre tento projekt som si nechal vyrobiť vlastnú dosku plošných spojov so štyrmi vrstvami: GND, napájacou a dvoma dátovými vrstvami.

Najdôležitejšou časťou softvéru je **PID regulácia**, ktorá umožňuje dronu dosiahnuť a udržať požadovanú orientáciu. Na úpravu PID koeficientov v reálnom čase využívam Bluetooth terminálovú aplikáciu. Táto aplikácia zároveň umožňuje získať aktuálny stav batérie.

Schéma zapojení:





Popis kódu:

```
#include "MPU6050_6Axis_MotionApps20.h"

#include "Wire.h"

#include <HardwareSerial.h>

#include "MPU6050.h"

#include <BluetoothSerial.h>



MPU6050 mpu;

MPU6050 accelgyro;

HardwareSerial IBusSerial(2); // Using UART2 (Serial2)

BluetoothSerial SerialBT;


#define RX_PIN 16 // Pin connected to FS-IA10B I-Bus signal

#define INTERRUPT_PIN 2  // use pin 2 on Arduino Uno & most boards

#define LED_PIN 12 // (Arduino is 13, Teensy is 11, Teensy++ is 6)
```

```
bool blinkState = false;

// MPU control/status vars
bool dmpReady = false;  // set true if DMP init was successful
uint8_t mpuIntStatus;   // holds actual interrupt status byte from MPU
uint8_t devStatus;      // return status after each device operation (0 = success, !0 = error)
uint16_t packetSize;    // expected DMP packet size (default is 42 bytes)
uint16_t fifoCount;     // count of all bytes currently in FIFO
uint8_t fifoBuffer[64]; // FIFO storage buffer

// orientation/motion vars
Quaternion q;           // [w, x, y, z]         quaternion container
VectorFloat gravity;    // [x, y, z]            gravity vector
float euler[3];         // [psi, theta, phi]    Euler angle container
float ypr[3];           // [yaw, pitch, roll]   yaw/pitch/roll container and gravity vector
// packet structure for InvenSense teapot demo
uint8_t teapotPacket[14] = { '$', 0x02, 0,0, 0,0, 0,0, 0,0, 0x00, 0x00, '\r', '\n' };
volatile bool mpuInterrupt = false;     // indicates whether MPU interrupt pin has gone high

uint8_t ibusData[32];
int ch1, ch2, ch3, ch4, ch10;

int16_t ax, ay, az;
int16_t gx, gy, gz;

int LF = 0;
int RF = 0;
int LR = 0;
int RR = 0;

float P = 40; //coefficients for bluetooth tuning
float I = 0;
float D = 40;
float T = 18;
float Y = 5;
float S = 2;

bool boolKill = 0; //safety features
```

```
bool arm=0;
bool latch=0;

int YAW = 0;
int PITCH = 0;
int ROLL = 0;

int frequency_pwm = 50;
int resolution_pwm = 12;
int LFpin = 18;  //definition of pins for escs
int LRpin = 0;
int RFpin = 5;
int RRpin = 4;

const int treshold_voltage = 1470;
float voltage=0;

const int voltagePin = 34;

float I_PITCH = 0;          // Accumulated integral value
float I_ROLL = 0;
unsigned long prevTime = 0;   // Previous time in milliseconds
unsigned long currTime = 0;   // Current time in milliseconds

void dmpDataReady() {
  mpuInterrupt = true;
}

void setup() {
  digitalWrite(12, HIGH);
  delay(1000);
  float voltage=analogRead(voltagePin); //measures the battery voltage
  if (voltage > treshold_voltage) digitalWrite(12, HIGH);

  Wire.begin();
  Wire.setClock(400000); // 400kHz I2C clock. Comment this line if having compilation difficulties
  Serial.begin(115200);
  SerialBT.begin("ProjectX"); // Bluetooth device name
```

```
Serial.println("The device started, now you can pair it with bluetooth!");


// initialize device
Serial.println(F("Initializing I2C devices..."));
mpu.initialize();
pinMode(INTERRUPT_PIN, INPUT);
// verify connection
Serial.println(mpu.testConnection() ? F("MPU6050 connection successful") : F("MPU6050 connection failed"));
// load and configure the DMP
devStatus = mpu.dmpInitialize();
// supply your own gyro offsets here, scaled for min sensitivity
mpu.setXGyroOffset(219);
mpu.setYGyroOffset(-30);
mpu.setZGyroOffset(28);
mpu.setZAccelOffset(1788); // 1688 factory default for my test chip
// make sure it worked (returns 0 if so)
if (devStatus == 0) {
  // Calibration Time: generate offsets and calibrate our MPU6050
  mpu.CalibrateAccel(6);
  mpu.CalibrateGyro(6);
  mpu.PrintActiveOffsets();
  // turn on the DMP, now that it's ready
  Serial.println(F("Enabling DMP..."));
  mpu.setDMPEnabled(true);
  // enable Arduino interrupt detection
  attachInterrupt(digitalPinToInterrupt(INTERRUPT_PIN), dmpDataReady, RISING);
  mpuIntStatus = mpu.getIntStatus();
  // set our DMP Ready flag so the main loop() function knows it's okay to use it
  Serial.println(F("DMP ready! Waiting for first interrupt..."));
  dmpReady = true;
  // get expected DMP packet size for later comparison
  packetSize = mpu.dmpGetFIFOPacketSize();
} else {
  Serial.print(F("DMP Initialization failed (code "));
  Serial.print(devStatus);
  Serial.println(F(")"));
}
```

```
  IBusSerial.begin(115200, SERIAL_8N1, RX_PIN, -1); // RX only on Serial2
  Serial.println("FlySky I-Bus Receiver Started...");

  ledcAttach(LFpin, frequency_pwm, resolution_pwm); //attaching PWM pins for escs
  ledcAttach(LRpin, frequency_pwm, resolution_pwm);
  ledcAttach(RFpin, frequency_pwm, resolution_pwm);
  ledcAttach(RRpin, frequency_pwm, resolution_pwm);

  // configure LED for output
  pinMode(LED_PIN, OUTPUT);
  digitalWrite(LED_PIN, HIGH);
}

void loop() {
  currTime = millis();        // Get the current time
  float dt = (currTime - prevTime) / 1000.0;  // Convert ms to seconds
  prevTime = currTime;
  // if programming failed, don't try to do anything
  if (!dmpReady) return;
  // read a packet from FIFO
  if (mpu.dmpGetCurrentFIFOPacket(fifoBuffer)) { // Get the Latest packet
   // display Euler angles in degrees
   mpu.dmpGetQuaternion(&q, fifoBuffer);
   mpu.dmpGetGravity(&gravity, &q);
   mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
   accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

   //Serial.println("pr:\t");
   YAW=ypr[0]* 180/M_PI;
   //Serial.print(YAW);
   //Serial.print("\t");
   PITCH=ypr[2]* 180/M_PI;
   // Serial.print(PITCH);
   // Serial.print("\t");
   ROLL=-ypr[1]* 180/M_PI;
   // Serial.print(ROLL);
```

```
  // Serial.print("a/g:\t");
  // Serial.print(ax); Serial.print("\t");
  // Serial.print(ay); Serial.print("\t");
  // Serial.print(az); Serial.print("\t");
  // Serial.print(gx); Serial.print("\t");
  // Serial.print(gy); Serial.print("\t");
  // Serial.println(gz);
}

I_PITCH += PITCH * dt; //integrating the PITCH and ROLL
I_ROLL += ROLL * dt;
// Serial.print(I_PITCH);
// Serial.print(I_ROLL);

if (SerialBT.available()) {    //section for bluetooth communication
char incomingChar = SerialBT.read();

switch (incomingChar) {
  case 'P':
    if (SerialBT.available()) {
      char incomingChar = SerialBT.read();
      if(incomingChar=='|'){
        P=0;
        while((SerialBT.available())){
          char incomingChar = SerialBT.read();
          if(incomingChar=='\n') {break;}
          P = 10*P + incomingChar - '0';
        }
        Serial.println(P);
      }
    }
    break;
  case 'I':
    if (SerialBT.available()) {
      char incomingChar = SerialBT.read();
      if(incomingChar=='|'){
        I=0;
        while((SerialBT.available())){
```

```
        char incomingChar = SerialBT.read();
        if(incomingChar=='\n') {break;}
        I = 10*I + incomingChar - '0';
      }
      Serial.println(I);
    }
  }
  break;
case 'D':
  if (SerialBT.available()) {
    char incomingChar = SerialBT.read();
    if(incomingChar=='|'){
      D=0;
      while((SerialBT.available())){
        char incomingChar = SerialBT.read();
        if(incomingChar=='\n') {break;}
        D = 10*D + incomingChar - '0';
      }
      Serial.println(D);
    }
  }
  break;
case 'T':
  if (SerialBT.available()) {
    char incomingChar = SerialBT.read();
    if(incomingChar=='|'){
      T=0;
      while((SerialBT.available())){
        char incomingChar = SerialBT.read();
        if(incomingChar=='\n') {break;}
        T = 10*T + incomingChar - '0';
      }
      Serial.println(T);
    }
  }
  break;
case 'S':
  if (SerialBT.available()) {
```

```
      char incomingChar = SerialBT.read();
      if(incomingChar=='|'){
        S=0;
        while((SerialBT.available())){
          char incomingChar = SerialBT.read();
          if(incomingChar=='\n') {break;}
          S = 10*S + incomingChar - '0';
        }
        Serial.println(S);
      }
    }
    break;
  case 'Y':
    if (SerialBT.available()) {
      char incomingChar = SerialBT.read();
      if(incomingChar=='|'){
        Y=0;
        while((SerialBT.available())){
          char incomingChar = SerialBT.read();
          if(incomingChar=='\n') {break;}
          Y = 10*Y + incomingChar - '0';
        }
        Serial.println(Y);
      }
    }
    break;
  case 'B':
    {SerialBT.print("Battery voltage: ");
    float voltage_out = voltage/106.7/4;
    SerialBT.println(voltage_out);
    }
    break;
  default:
    SerialBT.println("Invalid command");
    Serial.println("Invalid command");
    Serial.println(incomingChar);
    break;
}
```

```
}

  //part for communication with RF receiver
  // Check if data is available on Serial2 (I-Bus)
if (IBusSerial.available()) {
  // Read I-Bus data into the buffer (assuming 32 bytes max per frame)
  int index = 0;
  while (IBusSerial.available() && index < sizeof(ibusData)) {
    ibusData[index++] = IBusSerial.read();
  }
  // Process I-Bus data
  if (index == 32) { // 32 bytes is a complete frame
    // Extract channel data (starting after the header)
    // Header should be either 0x20 or 0x40 depending on I-Bus protocol
    if (ibusData[0] == 0x20 || ibusData[0] == 0x40) {
      for (int i = 2; i < index - 2; i += 2) { // Start at byte 2, step by 2
        // Combine two bytes to get a 16-bit channel value (Little-endian)
        uint16_t channelValue = ibusData[i] | (ibusData[i + 1] << 8);

        // FlySky I-Bus values should range between ~1000 and 2000
        if (channelValue >= 1000 && channelValue <= 2000) {
          // Serial.print(" Ch");
          // Serial.print((i - 2) / 2 + 1); // Channel number starts from 1
          // Serial.print(": ");
          // Serial.print(channelValue);
          if ((i - 2) / 2 + 1==1) {
            ch1=channelValue;
          }else if ((i - 2) / 2 + 1==2) {
            ch2=channelValue;
          }else if ((i - 2) / 2 + 1==3) {
            ch3=channelValue;
          }else if ((i - 2) / 2 + 1==4) {
            ch4=channelValue;
          }else if ((i - 2) / 2 + 1==10) {
            ch10=channelValue;
          }
        }
      }
```

```
    }
   }
   // Serial.println();
   // Serial.println(ch1);
   // Serial.println(ch2);
   // Serial.println(ch3);
   Serial.println(ch4);
   //Serial.println(ch10);
  }

//safety features, arm button and over angle cutoff
  if (ch10==2000 && latch == 0) {
   arm=1;
  }else if(ch10==2000 && latch==1){
   boolKill=0;
   arm=0;
  }else if(ch10 != 2000 && latch==1){
   latch=0;
   boolKill=0;
   arm=0;
  }else{
   boolKill=0;
   arm=0;
  }

  if((abs(ROLL)<45 && abs(PITCH)<45) && arm==1 && latch==0){
   boolKill=1;
  }else if(arm==1 && latch==0){
   latch==1;
   boolKill=0;
  }else{
   boolKill=0;
  }
  //Serial.println(boolKill);

  // Serial.println("pr:\t");
  // Serial.print(PITCH);
  // Serial.print("\t");
```

```
// Serial.print(ROLL);

//mapping values from sticks

float throttle = map(ch3, 1000, 2000, 0, 200);

float Ystick = map(ch4, 1000, 2000, -10, 10);

float Pstick = map(ch2, 1000, 2000, -10, 10);

float Rstick = map(ch1, 1000, 2000, -10, 10);

//equations for calculating the speed of motors, using PID control

LF=boolKill*(T * throttle + 0.1 *P * (ROLL + Rstick * S) + 0.1 * P * (PITCH - Pstick * S) + 0.001 * D * gy + 0.001 * D * gx + I * I_PITCH + I * I_ROLL + Y * Ystick);

LR=boolKill*(T * throttle + 0.1 *P * (ROLL + Rstick * S) - 0.1 * P * (PITCH - Pstick * S) + 0.001 * D * gy - 0.001 * D * gx - I * I_PITCH + I * I_ROLL - Y * Ystick);

RF=boolKill*(T * throttle - 0.1 *P * (ROLL + Rstick * S) + 0.1 * P * (PITCH - Pstick * S) - 0.001 * D * gy + 0.001 * D * gx + I * I_PITCH - I * I_ROLL - Y * Ystick);

RR=boolKill*(T * throttle - 0.1 *P * (ROLL + Rstick * S) - 0.1 * P * (PITCH - Pstick * S) - 0.001 * D * gy - 0.001 * D * gx - I * I_PITCH - I * I_ROLL + Y * Ystick);

// Serial.print("\t");

// Serial.print(LF);

// Serial.print("\t");

// Serial.print(LR);

// Serial.print("\t");

// Serial.print(RF);

// Serial.print("\t");

// Serial.print(RR);

// Serial.print("\t");

// Serial.println();


LF = constrain(LF, 0, 4095);

LR = constrain(LR, 0, 4095);

RF = constrain(RF, 0, 4095);

RR = constrain(RR, 0, 4095);


LF = map(LF, 0, 4095, 205, 410);//max 410

LR = map(LR, 0, 4095, 205, 410);//max 410

RF = map(RF, 0, 4095, 205, 410);//max 410

RR = map(RR, 0, 4095, 205, 410);//max 410


//outputs PWM for escs

ledcWrite(LFpin, LF);

ledcWrite(LRpin, LR);
```

```
  ledcWrite(RFpin, RF);
  ledcWrite(RRpin, RR);

  //measures the battry voltage
  voltage=analogRead(voltagePin);
  // Serial.println("\t voltage");
  // Serial.print(voltage);
  if (voltage < treshold_voltage){//overiť 680
    digitalWrite(12, HIGH);
  }else {
    digitalWrite(12, LOW);
  }
}
```