**Syrian Scientific Research**

**Project Report: “Night Vision Reconnaissance UAV”**

**Prepared by: Mahmoud Hassan**

**Supervised by: Dr. Samir Ali Raqia**

**2015**

**Codes**

1. PIC18F46K22 Firmware (XC8/C)

// main.c

#include <xc.h>

#include <stdint.h>

#include <stdbool.h>

#include <string.h>

// CONFIGURATION BITS

#pragma config PLLDIV = 2 // 8 MHz crystal / 2 = 4 MHz PLL input

#pragma config CPUDIV = OSC1\_PLL2 // CPU system clock = 64 MHz (16MHz \* 4 PLL / 2)

#pragma config FOSC = HSPLL // High-speed oscillator with PLL

#pragma config WDT = OFF // Watchdog Timer disabled

#pragma config LVP = OFF // Low-voltage programming disabled

#pragma config PBADEN = OFF // PORTB<4:0> pins are digital I/O

#define \_XTAL\_FREQ 64000000UL

// UART for ESP8266

void UART\_Init(void) {

TRISCbits.TRISC6 = 0; // TX pin

TRISCbits.TRISC7 = 1; // RX pin

SPBRG = 34; // Baud = 115200 @64MHz (BRG16=0, BRGH=1)

TXSTAbits.BRGH = 1;

BAUDCONbits.BRG16 = 0;

RCSTAbits.SPEN = 1; // Enable serial port

TXSTAbits.TXEN = 1; // Enable transmitter

RCSTAbits.CREN = 1; // Enable continuous receive

}

// I²C (MPU6050)

void I2C\_Init(void) {

TRISCbits.TRISC3 = 1; // SCL

TRISCbits.TRISC4 = 1; // SDA

SSPADD = (uint8\_t)((\_XTAL\_FREQ/(4UL\*100000UL))-1);

SSPSTAT = 0x80; // Slew rate disabled

SSPCON1 = 0x28; // I2C Master mode

}

void I2C\_Start(void) {

SSPCON2bits.SEN = 1;

while(SSPCON2bits.SEN);

}

void I2C\_Stop(void) {

SSPCON2bits.PEN = 1;

while(SSPCON2bits.PEN);

}

void I2C\_Write(uint8\_t data) {

SSPBUF = data;

while(SSPSTATbits.BF);

while(SSPCON2bits.ACKSTAT);

}

uint8\_t I2C\_Read(bool ack) {

uint8\_t received;

SSPCON2bits.RCEN = 1;

while(!SSPSTATbits.BF);

received = SSPBUF;

SSPCON2bits.ACKDT = ack ? 0 : 1; // ACK =0, NACK=1

SSPCON2bits.ACKEN = 1;

while(SSPCON2bits.ACKEN);

return received;

}

uint16\_t MPU\_Read(uint8\_t reg) {

uint16\_t val;

I2C\_Start();

I2C\_Write(0xD0); // SLA+W

I2C\_Write(reg);

I2C\_Start();

I2C\_Write(0xD1); // SLA+R

val = ((uint16\_t)I2C\_Read(1) << 8);

val |= ((uint16\_t)I2C\_Read(0));

I2C\_Stop();

return val;

}

// SPI for Thermal & Night-Vision Camera

void SPI\_Init(void) {

TRISCbits.TRISC5 = 0; // SDO

TRISCbits.TRISC4 = 1; // SDI

TRISCbits.TRISC3 = 0; // SCK

SSPSTAT = 0x40; // SPI data sampled at end

SSPCON1 = 0x21; // SPI Master, Fosc/16

}

uint8\_t SPI\_Transfer(uint8\_t data) {

SSPBUF = data;

while(!SSPSTATbits.BF);

return SSPBUF;

}

// Send raw buffer over UART to ESP8266

void UART\_Send(uint8\_t \*buf, uint16\_t len) {

for(uint16\_t i=0; i<len; i++) {

while(!TXSTAbits.TRMT);

TXREG = buf[i];

}

}

// Send AT+CIPSEND and raw packet

void ESP\_SendPacket(uint8\_t \*buf, uint16\_t len) {

char at[32];

uint16\_t cmdlen = sprintf(at, "AT+CIPSEND=%u\r\n", len);

UART\_Send((uint8\_t\*)at, cmdlen);

\_\_delay\_ms(50);

UART\_Send(buf, len);

}

// Motor control (ESC via PWM)

void PWM\_Init(void) {

// Use CCP1-CCP4 for motor PWM (Fosc/4 Timer2)

TRISCbits.TRISC2 = 0; // CCP1

TRISCbits.TRISC1 = 0; // CCP2

TRISEbits.PDE = 0; // CCP3 (on RE)

TRISEbits.PRE = 0; // CCP4 (on RE)

PR2 = 249; // PWM freq = 50Hz -> PR2 = (64MHz/(4\*4\*50Hz)) -1 = 249

T2CON = 0x04; // Timer2 on, prescaler 1

CCP1CON = 0x0C; // PWM mode

CCP2CON = 0x0C;

CCP3CON = 0x0C;

CCP4CON = 0x0C;

}

void SetMotor(uint8\_t id, uint16\_t pulse) {

// pulse range: 1000–2000us mapped to 0–1023 duty

uint16\_t duty = (uint32\_t)pulse \* 1023 / 20000;

switch(id) {

case 1: CCPR1L = duty>>2; CCP1CON &= 0xCF; CCP1CON |= (duty & 0x03)<<4; break;

case 2: CCPR2L = duty>>2; CCP2CON &= 0xCF; CCP2CON |= (duty & 0x03)<<4; break;

case 3: CCPR3L = duty>>2; CCP3CON &= 0xCF; CCP3CON |= (duty & 0x03)<<4; break;

case 4: CCPR4L = duty>>2; CCP4CON &= 0xCF; CCP4CON |= (duty & 0x03)<<4; break;

}

}

// Main

void main(void) {

OSCCON = 0x70; // 16MHz internal, PLL=on -> 64MHz

UART\_Init();

I2C\_Init();

SPI\_Init();

PWM\_Init();

// Configure ESP8266 in softAP mode etc.

UART\_Send((uint8\_t\*)"AT+CWMODE=2\r\n", 13);

\_\_delay\_ms(200);

UART\_Send((uint8\_t\*)"AT+CWSAP=\"DRONE\_NET\",\"pass123\",5,0\r\n", 37);

\_\_delay\_ms(200);

UART\_Send((uint8\_t\*)"AT+CIPMUX=1\r\n", 13);

\_\_delay\_ms(200);

UART\_Send((uint8\_t\*)"AT+CIPSERVER=1,8000\r\n", 19);

\_\_delay\_ms(200);

uint8\_t buf[512];

while(1) {

// 1) Read IMU

int16\_t ax = (int16\_t)MPU\_Read(0x3B);

int16\_t ay = (int16\_t)MPU\_Read(0x3D);

int16\_t az = (int16\_t)MPU\_Read(0x3F);

// 2) Control law (placeholder PID)

// ... compute thrusts T1..T4 ...

uint16\_t T[4] = {1500,1500,1500,1500};

for(uint8\_t i=0; i<4; i++) {

SetMotor(i+1, T[i]);

}

// 3) Receive command from ESP8266 over UART

if(PIR1bits.RCIF) {

uint8\_t c = RCREG;

// parse and adjust T[] or gimbal angles

}

// 4) Capture one line of thermal data over SPI

// (pseudo: we assume camera provides 64 bytes per row)

for(uint8\_t i=0; i<64; i++) {

buf[i] = SPI\_Transfer(0x00);

}

ESP\_SendPacket(buf, 64);

\_\_delay\_ms(20); // ~50Hz

}

}

## 2. C# WinForms Desktop Application

Solution: **DroneControlApp** Target: .NET 6, WinForms

### 2.1 Program Entry (Program.cs)

// Program.cs

using System;

using System.Windows.Forms;

namespace DroneControlApp

{

static class Program

{

[STAThread]

static void Main()

{

Application.SetHighDpiMode(HighDpiMode.SystemAware);

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new MainForm());

}

}

}

2.2 DroneLink Class (DroneLink.cs)

// DroneLink.cs

using System;

using System.Net.Sockets;

using System.Threading.Tasks;

namespace DroneControlApp

{

public class DroneLink : IDisposable

{

private UdpClient \_cmdClient;

private TcpClient \_videoClient;

private NetworkStream \_videoStream;

private readonly string \_ip;

private readonly int \_cmdPort;

private readonly int \_videoPort;

public DroneLink(string ip, int cmdPort = 9000, int videoPort = 8000)

{

\_ip = ip;

\_cmdPort = cmdPort;

\_videoPort = videoPort;

}

public async Task ConnectAsync()

{

\_cmdClient = new UdpClient();

\_cmdClient.Connect(\_ip, \_cmdPort);

\_videoClient = new TcpClient();

await \_videoClient.ConnectAsync(\_ip, \_videoPort);

\_videoStream = \_videoClient.GetStream();

}

public void SendCommand(string cmd)

{

if (\_cmdClient == null) return;

var data = System.Text.Encoding.ASCII.GetBytes(cmd + "\n");

\_cmdClient.Send(data, data.Length);

}

public async Task<byte[]> ReceiveFrameAsync()

{

// First 4 bytes = frame length

byte[] header = new byte[4];

int read = 0;

while (read < 4)

{

read += await \_videoStream.ReadAsync(header, read, 4 - read);

}

int len = BitConverter.ToInt32(header, 0);

byte[] frame = new byte[len];

read = 0;

while (read < len)

{

read += await \_videoStream.ReadAsync(frame, read, len - read);

}

return frame;

}

public void Dispose()

{

\_cmdClient?.Close();

\_videoClient?.Close();

\_videoStream?.Dispose();

}

}

}

### 2.3 MainForm & UI (MainForm.cs + MainForm.Designer.cs)

#### MainForm.cs

// MainForm.cs

using System;

using System.Drawing;

using System.IO;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace DroneControlApp

{

public partial class MainForm : Form

{

private DroneLink \_link;

private bool \_streaming;

public MainForm()

{

InitializeComponent();

}

private async void MainForm\_Load(object sender, EventArgs e)

{

\_link = new DroneLink("192.168.4.1");

await \_link.ConnectAsync();

StartVideoStream();

}

private async void StartVideoStream()

{

\_streaming = true;

while (\_streaming)

{

try

{

var bytes = await \_link.ReceiveFrameAsync();

using var ms = new MemoryStream(bytes);

var img = Image.FromStream(ms);

pictureBoxVideo.Image?.Dispose();

pictureBoxVideo.Image = new Bitmap(img);

}

catch

{

// handle errors or disconnect

\_streaming = false;

}

}

}

private void btnTakeOff\_Click(object sender, EventArgs e)

{

\_link.SendCommand("CMD:TAKEOFF");

}

private void btnLand\_Click(object sender, EventArgs e)

{

\_link.SendCommand("CMD:LAND");

}

private void btnUp\_Click(object sender, EventArgs e)

{

\_link.SendCommand("CMD:UP");

}

private void btnDown\_Click(object sender, EventArgs e)

{

\_link.SendCommand("CMD:DOWN");

}

private void btnLeft\_Click(object sender, EventArgs e)

{

\_link.SendCommand("CMD:LEFT");

}

private void btnRight\_Click(object sender, EventArgs e)

{

\_link.SendCommand("CMD:RIGHT");

}

private void MainForm\_FormClosing(object sender, FormClosingEventArgs e)

{

\_streaming = false;

\_link?.SendCommand("CMD:LAND");

\_link?.Dispose();

}

}

}

MainForm.Designer.cs

// MainForm.Designer.cs

namespace DroneControlApp

{

partial class MainForm

{

private System.ComponentModel.IContainer components = null;

private System.Windows.Forms.PictureBox pictureBoxVideo;

private System.Windows.Forms.Button btnTakeOff;

private System.Windows.Forms.Button btnLand;

private System.Windows.Forms.Button btnUp;

private System.Windows.Forms.Button btnDown;

private System.Windows.Forms.Button btnLeft;

private System.Windows.Forms.Button btnRight;

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

components.Dispose();

base.Dispose(disposing);

}

private void InitializeComponent()

{

this.pictureBoxVideo = new System.Windows.Forms.PictureBox();

this.btnTakeOff = new System.Windows.Forms.Button();

this.btnLand = new System.Windows.Forms.Button();

this.btnUp = new System.Windows.Forms.Button();

this.btnDown = new System.Windows.Forms.Button();

this.btnLeft = new System.Windows.Forms.Button();

this.btnRight = new System.Windows.Forms.Button();

((System.ComponentModel.ISupportInitialize)(this.pictureBoxVideo)).BeginInit();

this.SuspendLayout();

//

// pictureBoxVideo

//

this.pictureBoxVideo.Location = new System.Drawing.Point(12, 12);

this.pictureBoxVideo.Name = "pictureBoxVideo";

this.pictureBoxVideo.Size = new System.Drawing.Size(640, 480);

this.pictureBoxVideo.SizeMode = System.Windows.Forms.PictureBoxSizeMode.StretchImage;

this.pictureBoxVideo.TabIndex = 0;

this.pictureBoxVideo.TabStop = false;

//

// btnTakeOff

//

this.btnTakeOff.Location = new System.Drawing.Point(670, 20);

this.btnTakeOff.Name = "btnTakeOff";

this.btnTakeOff.Size = new System.Drawing.Size(100, 30);

this.btnTakeOff.TabIndex = 1;

this.btnTakeOff.Text = "Take Off";

this.btnTakeOff.UseVisualStyleBackColor = true;

this.btnTakeOff.Click += new System.EventHandler(this.btnTakeOff\_Click);

//

// btnLand

//

this.btnLand.Location = new System.Drawing.Point(670, 60);

this.btnLand.Name = "btnLand";

this.btnLand.Size = new System.Drawing.Size(100, 30);

this.btnLand.TabIndex = 2;

this.btnLand.Text = "Land";

this.btnLand.UseVisualStyleBackColor = true;

this.btnLand.Click += new System.EventHandler(this.btnLand\_Click);

//

// btnUp

//

this.btnUp.Location = new System.Drawing.Point(720, 120);

this.btnUp.Name = "btnUp";

this.btnUp.Size = new System.Drawing.Size(50, 30);

this.btnUp.TabIndex = 3;

this.btnUp.Text = "↑";

this.btnUp.UseVisualStyleBackColor = true;

this.btnUp.Click += new System.EventHandler(this.btnUp\_Click);

//

// btnDown

//

this.btnDown.Location = new System.Drawing.Point(720, 200);

this.btnDown.Name = "btnDown";

this.btnDown.Size = new System.Drawing.Size(50, 30);

this.btnDown.TabIndex = 4;

this.btnDown.Text = "↓";

this.btnDown.UseVisualStyleBackColor = true;

this.btnDown.Click += new System.EventHandler(this.btnDown\_Click);

//

// btnLeft

//

this.btnLeft.Location = new System.Drawing.Point(670, 160);

this.btnLeft.Name = "btnLeft";

this.btnLeft.Size = new System.Drawing.Size(50, 30);

this.btnLeft.TabIndex = 5;

this.btnLeft.Text = "←";

this.btnLeft.UseVisualStyleBackColor = true;

this.btnLeft.Click += new System.EventHandler(this.btnLeft\_Click);

//

// btnRight

//

this.btnRight.Location = new System.Drawing.Point(770, 160);

this.btnRight.Name = "btnRight";

this.btnRight.Size = new System.Drawing.Size(50, 30);

this.btnRight.TabIndex = 6;

this.btnRight.Text = "→";

this.btnRight.UseVisualStyleBackColor = true;

this.btnRight.Click += new System.EventHandler(this.btnRight\_Click);

//

// MainForm

//

this.ClientSize = new System.Drawing.Size(840, 510);

this.Controls.Add(this.btnRight);

this.Controls.Add(this.btnLeft);

this.Controls.Add(this.btnDown);

this.Controls.Add(this.btnUp);

this.Controls.Add(this.btnLand);

this.Controls.Add(this.btnTakeOff);

this.Controls.Add(this.pictureBoxVideo);

this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedSingle;

this.Name = "MainForm";

this.Text = "Drone Control Station";

this.Load += new System.EventHandler(this.MainForm\_Load);

this.FormClosing += new System.Windows.Forms.FormClosingEventHandler(this.MainForm\_FormClosing);

((System.ComponentModel.ISupportInitialize)(this.pictureBoxVideo)).EndInit();

this.ResumeLayout(false);

}

}

}