**Приложение A. Текст программы**

## АННОТАЦИЯ

В данном программном документе приведен текст программы встроенного приложения с предиктивной коррекцией ошибок управления.

В разделе «Текст программы» указано назначение программы, краткая характеристика области применения программы, описание модулей и их программный код.

СОДЕРЖАНИЕ

[1. ТЕКСТ ПРОГРАММЫ 3](#_Toc67731117)

[1.1. Наименование программы 3](#_Toc67731118)

[1.2. Область применения программы 3](#_Toc67731119)

[1.3. Модули 3](#_Toc67731120)

[1.4. Код программы 3](#_Toc67731121)

## ТЕКСТ ПРОГРАММЫ

### Наименование программы

Наименование – программа встроенного приложения с предиктивной коррекцией ошибок управления.

### Область применения программы

Программа должна эксплуатироваться в составе программно-аппаратного комплекса в виде платформы-носителя с универсальным интерфейсом связи «MasterLink». Конечными пользователями программы должны являться сотрудники с допуском работы на промышленном оборудовании с автоматическим управлением подвижными частями.

### Модули

Таблица 1 - Модули.

| № | Название модуля | Описание модуля | Размер модуля | Кол-во строк |
| --- | --- | --- | --- | --- |
| 1 | PlatformMain.cpp | Модуль основной программы | 1 кб | 48 |
| 2 | Platform.h | Заголовочный файл библиотеки Platform | 8 кб | 116 |
| 3 | Platform.cpp | Модуль логики библиотеки Platform | 15 кб | 542 |
| 4 | Arduino.h | Заголовочный файл библиотеки Arduino | 10 кб | 260 |

### Код программы

* + 1. PlatformMain.cpp

#include "Platform.h"

Platform platform;

void setup() {

pinMode(13, OUTPUT);

Serial.begin(115200); //Debug

Serial1.begin(9600); //GPS

platform.begin("testPlatf", "8tegqHu6VZ");

platform.initControlUARTData(platform, 34); //57600

platform.initMPU();

}

void loop() {

//while(1) { //Speed-up bug

//PORTB |= (1 << 7); //13 test square generator

//PORTB &= ~ (1 << 7); //13

//if (millis() % 100 == 0) {

//delay(100);

//platform.getMPUData();

//}

//EVERY\_MS(300) {

//test = !test;

//}

//digitalWrite(13, test);

//}

//platform.startBench();

//delay(500);

platform.getMPUData();

platform.getGPSData(&Serial1);

//platform.stopBench(&Serial);

}

/\* TODO

Получение вольтажа батареи

Заполение структуры данными MPU

Конфигуратор GPIO, запись напрямую через текстовые команды и работа с интерфейсами через data в dataIncome

Подумать над структурой бибилотеки

uart провод 115200 23,944 ms

uart радио 9600 1,1 ms примерно

%:f,100,s,10,1,0,0,0,1;

\*/

* + 1. Platform.h

#pragma once

#include <Arduino.h>

#include <avr/interrupt.h>

#include <Wire.h>

#define \_LIB\_VERSION 1.0

#define DEBUGGYRO false

#define DEBUGACC false

#define DEBUGUART false

#define DEBUGGPS false

#define MPU6050\_ADDRESS 0x68

#define BACKWARD 0 // определяем значение движения назад

#define FORWARD 1 // определяем значение движения вперед

#define LEFT 2 // определяем значение вращения против часовой стрелки

#define RIGHT 3 // определяем значение вращения по часовой стрелке

#define FORWARDLEFT 4 // определяем значение движения вперед и налево

#define FORWARDRIGHT 5 // определяем значение движения вперед и вправо

#define BACKWARDLEFT 6 // определяем значение движения назад и налево

#define BACKWARDRIGHT 7 // определяем значение движения назад и вправо

#define BRAKE 1 // определяем значение резкого тормоза

#define STOP 0 // определяем значение остановки

#define FAST 0 // определяем значение резкого разгона

#define SLOW 1 // определяем значение плавного разгона

const uint8\_t inApin[2] = {7, 4}; // определяем выводы ключей A

const uint8\_t inBpin[2] = {8, 9}; // определяем выводы ключей B

const uint8\_t pwmpin[2] = {5, 6}; // определяем выводы ШИМ

const uint8\_t cspin[2] = {A2, A3}; // определяем выводы считывания тока

const uint8\_t enpin[2] = {A0, A1}; // определяем выводы состояния ключей AB. Ключи открываются, если притянуть к 0

struct DataIncome { // Структура данных, приходящих с ПК по UART

char move;

uint8\_t speed;

char value;

uint8\_t azimuthloc;

uint8\_t gpio1 = 0;

uint8\_t gpio2 = 0;

uint8\_t gpio3 = 0;

uint8\_t gpio4 = 0;

uint8\_t systemstatus = 0;

String data;

};

struct DataOutcome { // Структура данных, исходящих по UART

char move;

uint8\_t speed;

char value;

uint16\_t lcurr;

uint16\_t rcurr;

float accx;

float accy;

float accz;

float gyrox;

float gyroy;

float gyroz;

float magx;

float magy;

float magz;

String lan;

String lon;

uint8\_t vbat;

uint16\_t extid = 0;

uint8\_t extstatus = 0;

};

class Platform { // класс Platform

public:

DataIncome controlDataIn;

DataOutcome controlDataOut;

String GPSTimestamp = "";

String GPSLatitude = "";

String GPSLongitude = "";

//MPU6050 sensor

volatile float AccX, AccY, AccZ;

volatile float GyroX, GyroY, GyroZ;

volatile float AccErrorX, AccErrorY, GyroErrorX, GyroErrorY, GyroErrorZ;

volatile float Temperature;

volatile int MPU\_Calib\_Counter = 0;

volatile float AccDevider, GyroDevider = 0;

Platform();

void begin(String name, String key);

//Movements section

void makeMove(uint8\_t direction, uint8\_t speed, uint8\_t acceleration);

void brake(uint8\_t mode);

//Telemetry section

bool initControlUARTData(Platform platform, int baudrate);

void getControlUARTData(void);

uint8\_t readBatVoltage(); // Need to create method int

bool getGPSData(Stream\* \_serial);

void initMPU();

void getMPUData();

//Another useful functions

void startBench();

void stopBench(Stream\* \_serial);

float convertRawCoordinatesToDegrees(float RawDegrees);

void I2Cread(uint8\_t Address, uint8\_t Register, uint8\_t Nbytes, uint8\_t\* Data);

void I2CwriteByte(uint8\_t Address, uint8\_t Register, uint8\_t Data);

private:

String PlatformKey = ""; // Приватный ключ платформы

String PlatformName = ""; // Имя платформы

String stringUARTCommand = ""; // Переменная сбора принятых символов в строку

volatile bool startedUARTCommandRecieve; // переменная начала приема командных данных по uart

volatile uint8\_t indexUARTCommand = 0; // Индекс принятого аргумента командного режима

};

* + 1. Platform.cpp

#include "Platform.h"

#define cbi(sfr, bit) (\_SFR\_BYTE(sfr) &= ~\_BV(bit))

#define sbi(sfr, bit) (\_SFR\_BYTE(sfr) |= \_BV(bit))

Platform ptf; //Class instance call

Platform::Platform() {} // конструктор

void Platform::begin(String name, String key) {

PlatformName = name;

PlatformKey = key;

pinMode(7, OUTPUT); // выводы ключей A

pinMode(4, OUTPUT); // выводы ключей A

pinMode(8, OUTPUT); // выводы ключей B

pinMode(9, OUTPUT); // выводы ключей B

pinMode(5, OUTPUT); // выводы ШИМ

pinMode(6, OUTPUT); // выводы ШИМ

pinMode(A0, INPUT); // выводы датчиков тока

pinMode(A1, INPUT); // выводы датчиков тока

sbi(TCCR3A, COM3A1); //PWM

sbi(TCCR4A, COM4A1);

controlDataIn.systemstatus = 1;

}

void Platform::makeMove(uint8\_t direction, uint8\_t speed, uint8\_t acceleration) {

uint8\_t dividerForRightMotor = 0;

uint8\_t dividerForLeftMotor = 0;

PORTH &= ~ (1 << 4); //7, LOW A

PORTG &= ~ (1 << 5); //4, LOW A

PORTH &= ~ (1 << 5); //8, LOW B

PORTH &= ~ (1 << 6); //9, LOW B

switch (direction) {

case 0:

PORTH |= (1 << 4); //7, HIGH A

PORTH |= (1 << 6); //9, HIGH B

break;

case 1:

PORTG |= (1 << 5); //4, HIGH A

PORTH |= (1 << 5); //8, HIGH B

break;

case 2:

PORTH &= ~ (1 << 4); //7, LOW A

PORTH |= (1 << 5); //8, HIGH B

PORTG &= ~ (1 << 5); //4, LOW A

PORTH |= (1 << 6); //9, HIGH B

break;

case 3:

PORTH |= (1 << 4); //7, HIGH A

PORTH &= ~ (1 << 5); //8, LOW B

PORTG |= (1 << 5); //4, HIGH A

PORTH &= ~ (1 << 6); //9, LOW B

break;

case 4:

PORTG |= (1 << 5); //4, HIGH A

PORTH |= (1 << 5); //8, HIGH B

dividerForRightMotor = 5;

dividerForLeftMotor = 0; //Decrease left speed

break;

case 5:

PORTG |= (1 << 5); //4, HIGH A

PORTH |= (1 << 5); //8, HIGH B

dividerForRightMotor = 0; //Decrease right speed

dividerForLeftMotor = 5;

break;

case 6:

PORTH |= (1 << 4); //7, HIGH A

PORTH |= (1 << 6); //9, HIGH B

dividerForRightMotor = 5;

dividerForLeftMotor = 0; //Decrease left speed

break;

case 7:

PORTH |= (1 << 4); //7, HIGH A

PORTH |= (1 << 6); //9, HIGH B

dividerForRightMotor = 0; //Decrease right speed

dividerForLeftMotor = 5;

break;

}

/\*if(acceleration == 1) {

for(uint16\_t i = 0; i <= map(speed, 0, 100, 0, 255); i++) { //Не работает. работает. да...

analogWrite(pwmpin[0], i);

analogWrite(pwmpin[1], i);

delay(10);

}

}

else {\*/

OCR3A = map(speed<5?speed:speed-dividerForRightMotor, 0, 100, 0, 255); // set pwm duty

OCR4A = map(speed<5?speed:speed-dividerForLeftMotor, 0, 100, 0, 255);

//}

}

void Platform::brake(uint8\_t mode) {

if (mode == 1) {

OCR3A = 0; // set pwm duty

OCR4A = 0;

//Резко тормозим, замыкая обмотки

PORTH |= (1 << 4); //7, HIGH

PORTG |= (1 << 5); //4, HIGH

PORTH |= (1 << 5); //8, HIGH

PORTH |= (1 << 6); //9, HIGH

delay(50);

//Возвращаем ключи в нулевое состояние

PORTH &= ~ (1 << 4); //7, LOW

PORTG &= ~ (1 << 5); //4, LOW

PORTH &= ~ (1 << 5); //8, LOW

PORTH &= ~ (1 << 6); //9, LOW

}

else {

OCR3A = 0; // set pwm duty

OCR4A = 0;

//Мягко тормозим по инерции

PORTH &= ~ (1 << 4); //7, LOW

PORTG &= ~ (1 << 5); //4, LOW

PORTH &= ~ (1 << 5); //8, LOW

PORTH &= ~ (1 << 6); //9, LOW

}

}

//Telemetry section

bool Platform::initControlUARTData(Platform platform, int baudrate) {

UCSR2A = 1 << U2X1; //UCSR2A = 1 << U2X1 for 115200

// assign the baud\_setting, a.k.a. ubrr (USART Baud Rate Register)

/\* Set baud rate \*/

UBRR2H = baudrate >> 8;

UBRR2L = baudrate;

//Разрешение на прием и на передачу через USART, прерывания по поступлению и по опустошению

UCSR2B = (1 << RXCIE2) | (1 << TXCIE2) | (1 << RXEN2) | (1 << TXEN2);

UCSR2C = (1 << UCSZ21) | (1 << UCSZ20); //размер слова 8 разрядов

sei();

ptf = platform;

return true;

}

ISR(USART2\_RX\_vect) { //ISR UART2 handler

ptf.getControlUARTData();

}

void Platform::getControlUARTData(void) {

while ( !(UCSR2A & (1 << RXC2)) );

char incomingByte = UDR2; // обязательно ЧИТАЕМ входящий символ

if (startedUARTCommandRecieve) { // если приняли начальный символ (парсинг разрешён)

if (incomingByte != ',' && incomingByte != ';') { // если это не пробел И не конец

stringUARTCommand += incomingByte; // складываем в строку

} else { // если это пробел или ; конец пакета

switch (indexUARTCommand) {

case 0:

controlDataIn.move = stringUARTCommand[1];

break;

case 1:

controlDataIn.speed = stringUARTCommand.toInt();

break;

case 2:

controlDataIn.value = stringUARTCommand[0];

break;

case 3:

controlDataIn.azimuthloc = stringUARTCommand.toInt();

break;

case 4:

controlDataIn.gpio1 = stringUARTCommand.toFloat();

break;

case 5:

controlDataIn.gpio2 = stringUARTCommand.toFloat();

break;

case 6:

controlDataIn.gpio3 = stringUARTCommand.toFloat();

break;

case 7:

controlDataIn.gpio4 = stringUARTCommand.toFloat();

break;

case 8:

controlDataIn.systemstatus = stringUARTCommand.toInt();

break;

case 9:

controlDataIn.data = stringUARTCommand;

break;

}

stringUARTCommand = ""; // очищаем строку

indexUARTCommand++; // переходим к парсингу следующего элемента массива

}

}

if (incomingByte == '%') { // если это $

startedUARTCommandRecieve = true; // поднимаем флаг, что можно парсить

indexUARTCommand = 0; // сбрасываем индекс

stringUARTCommand = ""; // очищаем строку

}

if (incomingByte == ';') { // если таки приняли ; - конец парсинга

startedUARTCommandRecieve = false; // сброс

controlDataOut.move = controlDataIn.move; //Заполняем структуру и передаем её

controlDataOut.speed = controlDataIn.speed;

controlDataOut.value = controlDataIn.value;

controlDataOut.lcurr = analogRead(cspin[1]) \* 0.038; //Current in Amps

controlDataOut.rcurr = analogRead(cspin[0]) \* 0.038;

// controlDataOut.accx = AccX;

// controlDataOut.accy = AccY;

// controlDataOut.accz = AccZ;

// controlDataOut.gyrox = GyroX;

// controlDataOut.gyroy = GyroY;

// controlDataOut.gyroz = GyroZ;

controlDataOut.magx = 0;

controlDataOut.magy = 0;

controlDataOut.magz = 0;

controlDataOut.lan = GPSLatitude;

controlDataOut.lon = GPSLongitude;

controlDataOut.vbat = 0;

controlDataOut.extid = 0;

controlDataOut.extstatus = 0;

//Serial.println(ptf.controlDataOut.gyrox);

String outgoingDataString = "&:" + String(controlDataOut.move) + "," + String(controlDataOut.speed) + "," + String(controlDataOut.value) + "," + String(controlDataOut.lcurr) + "," + String(controlDataOut.rcurr) + "," + String(controlDataOut.accx) + "," + String(controlDataOut.accy) + "," + String(controlDataOut.accz) + "," + String(controlDataOut.gyrox) + "," + String(controlDataOut.gyroy) + "," + String(controlDataOut.gyroz) + "," + String(controlDataOut.magx) + "," + String(controlDataOut.magy) + "," + String(controlDataOut.magz) + "," + controlDataOut.lan + "," + controlDataOut.lon + "," + String(controlDataOut.vbat) + "," + String(controlDataOut.extid) + "," + String(controlDataOut.extstatus) + ";\r\n";

//String outgoingDataString = "&:" +PlatformName+"," + String(controlDataOut.move) + "," + String(controlDataOut.speed) + "," + String(controlDataOut.value) + "," + String(controlDataOut.lcurr) + "," + String(controlDataOut.rcurr) + "," + String(controlDataOut.accx) + "," + String(controlDataOut.accy) + "," + String(controlDataOut.accz) + "," + String(controlDataOut.gyrox) + "," + String(controlDataOut.gyroy) + "," + String(controlDataOut.gyroz) + "," + String(controlDataOut.magx) + "," + String(controlDataOut.magy) + "," + String(controlDataOut.magz) + "," + controlDataOut.lan + "," + controlDataOut.lon + "," + String(controlDataOut.vbat) + "," + String(controlDataOut.extid) + "," + String(controlDataOut.extstatus) + ";\r\n";

for (uint32\_t i = 0; i <= strlen(outgoingDataString.c\_str()); ++i) {

/\* Wait for empty transmit buffer \*/

while ( !( UCSR2A & (1 << UDRE2)) );

/\* Put data into buffer, sends the data \*/

UDR2 = outgoingDataString[i];

}

switch (controlDataIn.move) {

case 'f':

makeMove(FORWARD, controlDataIn.speed, (controlDataIn.value == 'f') ? FAST : SLOW);

break;

case 'b':

makeMove(BACKWARD, controlDataIn.speed, (controlDataIn.value == 'f') ? FAST : SLOW);

break;

case 'l':

makeMove(LEFT, controlDataIn.speed, (controlDataIn.value == 'f') ? FAST : SLOW);

break;

case 'r':

makeMove(RIGHT, controlDataIn.speed, (controlDataIn.value == 'f') ? FAST : SLOW);

break;

case 'a':

makeMove(FORWARDLEFT, controlDataIn.speed, (controlDataIn.value == 'f') ? FAST : SLOW);

break;

case 'c':

makeMove(FORWARDRIGHT, controlDataIn.speed, (controlDataIn.value == 'f') ? FAST : SLOW);

break;

case 'd':

makeMove(BACKWARDLEFT, controlDataIn.speed, (controlDataIn.value == 'f') ? FAST : SLOW);

break;

case 'e':

makeMove(BACKWARDRIGHT, controlDataIn.speed, (controlDataIn.value == 'f') ? FAST : SLOW);

break;

case 's':

brake(STOP);

break;

}

}

}

bool Platform::getGPSData(Stream\* \_serial) {

String stringGPS = "";

if (\_serial->available() > 0) {

stringGPS = \_serial->readStringUntil(13); //NMEA data ends with 'return' character, which is ascii(13)

stringGPS.trim(); // they say NMEA data starts with "$", but the Arduino doesn't think so.

//Serial.println(stringGPS); //All the raw sentences will be sent to monitor, if you want them, maybe to see the labels and data order.

//Start Parsing by finding data, put it in a string of character array, then removing it, leaving the rest of thes sentence for the next 'find'

if (stringGPS.startsWith("$GPGLL") || stringGPS.startsWith("$GLGLL") || stringGPS.startsWith("$GAGLL") || stringGPS.startsWith("$BDGLL") || stringGPS.startsWith("$GQGLL") || stringGPS.startsWith("$GNGLL")) { //I picked this sentence, you can pick any of the other labels and rearrange/add sections as needed.

//Serial.println(stringGPS); // display raw GLL data in Serial Monitor

// mine looks like this: "$GPGLL,4053.16598,N,10458.93997,E,224431.00,A,D\*7D"

//This section gets repeated for each delimeted bit of data by looking for the commas

//Find Lattitude is first in GLL sentence, other senetences have data in different order

int Pos = stringGPS.indexOf(','); //look for comma delimetrer

stringGPS.remove(0, Pos + 1); // Remove Pos+1 characters starting at index=0, this one strips off "$GPGLL" in my sentence

Pos = stringGPS.indexOf(','); //looks for next comma delimetrer, which is now the first comma because I removed the first segment

char Lat[Pos]; //declare character array Lat with a size of the dbit of data

for (int i = 0; i <= Pos - 1; i++) { // load charcters into array

Lat[i] = stringGPS.charAt(i);

}

//Serial.print(Lat); // display raw latitude data in Serial Monitor, I'll use Lat again in a few lines for converting

//repeating with a different char array variable

//Get Lattitude North or South

stringGPS.remove(0, Pos + 1);

Pos = stringGPS.indexOf(',');

char LatSide[Pos]; //declare different variable name

for (int i = 0; i <= Pos - 1; i++) {

LatSide[i] = stringGPS.charAt(i); //fill the array

//Serial.println(LatSide[i]); //display N or S

}

//convert the variable array Lat to degrees Google can use

float LatAsFloat = atof (Lat); //atof converts the char array to a float type

float LatInDeg;

if (LatSide[0] == char(78)) { //char(69) is decimal for the letter "N" in ascii chart

LatInDeg = convertRawCoordinatesToDegrees(LatAsFloat); //call the conversion funcion (see below)

}

if (LatSide[0] == char(83)) { //char(69) is decimal for the letter "S" in ascii chart

LatInDeg = -( convertRawCoordinatesToDegrees(LatAsFloat)); //call the conversion funcion (see below)

}

GPSLatitude = String(LatInDeg, 8); //TEMP SOLUTION

//Serial.println(LatInDeg, 15); //display value Google can use in Serial Monitor, set decimal point value high

//repeating with a different char array variable

//Get Longitude

stringGPS.remove(0, Pos + 1);

Pos = stringGPS.indexOf(',');

char Longit[Pos]; //declare different variable name

for (int i = 0; i <= Pos - 1; i++) {

Longit[i] = stringGPS.charAt(i); //fill the array

}

//Serial.print(Longit); //display raw longitude data in Serial Monitor

//repeating with a different char array variable

//Get Longitude East or West

stringGPS.remove(0, Pos + 1);

Pos = stringGPS.indexOf(',');

char LongitSide[Pos]; //declare different variable name

for (int i = 0; i <= Pos - 1; i++) {

LongitSide[i] = stringGPS.charAt(i); //fill the array

//Serial.println(LongitSide[i]); //display raw longitude data in Serial Monitor

}

//convert to degrees Google can use

float LongitAsFloat = atof (Longit); //atof converts the char array to a float type

float LongInDeg;

if (LongitSide[0] == char(69)) { //char(69) is decimal for the letter "E" in ascii chart

LongInDeg = convertRawCoordinatesToDegrees(LongitAsFloat); //call the conversion funcion (see below

}

if (LongitSide[0] == char(87)) { //char(87) is decimal for the letter "W" in ascii chart

LongInDeg = -(convertRawCoordinatesToDegrees(LongitAsFloat)); //call the conversion funcion (see below

}

GPSLongitude = String(LongInDeg, 8); //TEMP SOLUTION

//Serial.println(LongInDeg, 15); //display value Google can use in Serial Monitor, set decimal point value high

//repeating with a different char array variable

//Get TimeStamp - GMT

stringGPS.remove(0, Pos + 1);

Pos = stringGPS.indexOf(',');

char TimeStamp[Pos]; //declare different variable name

for (int i = 0; i <= Pos - 1; i++) {

TimeStamp[i] = stringGPS.charAt(i); //fill the array

}

GPSTimestamp = TimeStamp; //TEMP SOLUTION

//Serial.print(TimeStamp); //display raw longitude data in Serial Monitor, GMT

Serial.println("");

}

}

return true;

}

void Platform::initMPU() {

Wire.begin();

Wire.setClock(400000);

I2CwriteByte(MPU6050\_ADDRESS, 29, 0x06);// Set accelerometers low pass filter at 5Hz !

I2CwriteByte(MPU6050\_ADDRESS, 26, 0x06); // Set gyroscope low pass filter at 5Hz !

// Configure gyroscope range

I2CwriteByte(MPU6050\_ADDRESS, 27, 0x6B); GyroDevider = 131; //GYRO\_FULL\_SCALE\_250\_DPS !

//I2CwriteByte(MPU6050\_ADDRESS, 27, 0x08); GyroDevider = 65.5; //GYRO\_FULL\_SCALE\_500\_DPS

//I2CwriteByte(MPU6050\_ADDRESS, 27, 0x10); GyroDevider = 32.8; //GYRO\_FULL\_SCALE\_1000\_DPS

// I2CwriteByte(MPU6050\_ADDRESS, 27, 0x18); GyroDevider = 16.4; //GYRO\_FULL\_SCALE\_2000\_DPS

// Configure accelerometers range

I2CwriteByte(MPU6050\_ADDRESS, 28, 0x00); AccDevider = 16384; //ACC\_FULL\_SCALE\_2\_G !

//I2CwriteByte(MPU6050\_ADDRESS, 28, 0x08); AccDevider = 8192; //ACC\_FULL\_SCALE\_4\_G

//I2CwriteByte(MPU6050\_ADDRESS, 28, 0x10); AccDevider = 4096; //ACC\_FULL\_SCALE\_8\_G

//I2CwriteByte(MPU6050\_ADDRESS, 28, 0x18); AccDevider = 2048; //ACC\_FULL\_SCALE\_16\_G

while (MPU\_Calib\_Counter < 200) {

uint8\_t Buf[14];

I2Cread(MPU6050\_ADDRESS, 0x3B, 14, Buf);

//Get values from sensor

GyroX = -(Buf[0] << 8 | Buf[1]);

GyroY = -(Buf[2] << 8 | Buf[3]);

GyroZ = Buf[4] << 8 | Buf[5];

// Sum all readings

GyroErrorX = GyroErrorX + (GyroX / GyroDevider);

GyroErrorY = GyroErrorY + (GyroY / GyroDevider);

GyroErrorZ = GyroErrorZ + (GyroZ / GyroDevider);

MPU\_Calib\_Counter++;

}

//Divide the sum by 200 to get the error value

GyroErrorX = GyroErrorX / 200;

GyroErrorY = GyroErrorY / 200;

GyroErrorZ = GyroErrorZ / 200;

MPU\_Calib\_Counter = 0;

while (MPU\_Calib\_Counter < 200) {

uint8\_t Buf[14];

I2Cread(MPU6050\_ADDRESS, 0x3B, 14, Buf);

//Get values from sensor

AccX = (Buf[8] << 8 | Buf[9]) / AccDevider;

AccY = (Buf[10] << 8 | Buf[11]) / AccDevider;

AccZ = (Buf[12] << 8 | Buf[13]) / AccDevider;

// Sum all readings

AccErrorX = AccErrorX + ((atan((AccY) / sqrt(pow((AccX), 2) + pow((AccZ), 2))) \* 180 / PI));

AccErrorY = AccErrorY + ((atan(-1 \* (AccX) / sqrt(pow((AccY), 2) + pow((AccZ), 2))) \* 180 / PI));

MPU\_Calib\_Counter++;

}

//Divide the sum by 200 to get the error value

AccErrorX = AccErrorX / 200;

AccErrorY = AccErrorY / 200;

MPU\_Calib\_Counter = 0;

#if DEBUGGYRO || DEBUGACC

Serial.print(F("AccErrorX: "));

Serial.println(AccErrorX);

Serial.print(F("AccErrorY: "));

Serial.println(AccErrorY);

Serial.print(F("GyroErrorX: "));

Serial.println(GyroErrorX);

Serial.print(F("GyroErrorY: "));

Serial.println(GyroErrorY);

Serial.print(F("GyroErrorZ: "));

Serial.println(GyroErrorZ);

#endif

}

void Platform::getMPUData() {

uint8\_t Buf[14];

I2Cread(MPU6050\_ADDRESS, 0x3B, 14, Buf); // Read accelerometer and gyroscope

//Gyroscope

GyroX = (Buf[0] << 8 | Buf[1]) / GyroDevider;

GyroY = (Buf[2] << 8 | Buf[3]) / GyroDevider;

GyroZ = (Buf[4] << 8 | Buf[5]) / GyroDevider;

// Correct the outputs with the calculated error values

GyroX = GyroX + abs(GyroErrorX); // GyroErrorX ~(-0.56)

GyroY = GyroY + abs(GyroErrorY); // GyroErrorY ~(2)

GyroZ = GyroZ + abs(GyroErrorZ); // GyroErrorZ ~ (-0.8)

//Temperature

Temperature = (Buf[6] << 8 | Buf[7]) / 340.0 + 36.53;

// Accelerometer

AccX = (Buf[8] << 8 | Buf[9]) / AccDevider;

AccY = (Buf[10] << 8 | Buf[11]) / AccDevider;

AccZ = (Buf[12] << 8 | Buf[13]) / AccDevider;

// Display values

ptf.controlDataOut.accx = AccX;

ptf.controlDataOut.accy = AccY;

ptf.controlDataOut.accz = AccZ;

ptf.controlDataOut.gyrox = GyroX;

ptf.controlDataOut.gyroy = GyroY;

ptf.controlDataOut.gyroz = GyroZ;

//Serial.println(ptf.controlDataOut.gyrox);

// Gyroscope

#if DEBUGGYRO

Serial.print(F("GyroX: "));

Serial.println((int)GyroX, DEC);

Serial.print(F("GyroY: "));

Serial.println((int)GyroY, DEC);

Serial.print(F("GyroZ: "));

Serial.println((int)GyroZ, DEC);

Serial.println((int)Temperature, DEC);

#endif

// Accelerometer

#if DEBUGACC

Serial.print(F("AccX: "));

Serial.println(AccX, DEC);

Serial.print(F("AccY: "));

Serial.println(AccY, DEC);

Serial.print(F("AccZ: "));

Serial.println (AccZ, DEC);

#endif

}

//Another useful functions

void Platform::startBench() {

TCCR1A = 0x00; // выключаем

TCCR1B = 0x00; // выключаем

TCNT1 = 0x00; // сброс счётчика

TCCR1B = 0x01; // запустить таймер

}

void Platform::stopBench(Stream\* \_serial) {

TCCR1B = 0x00; // остановить таймер

uint32\_t count = TCNT1 - 2; // минус два такта на действия

\_serial->print("ticks: ");

\_serial->print(count);

\_serial->print(" ");

\_serial->print("time (us): ");

\_serial->println(count \* (float)(1000000.0f / F\_CPU), 4);

}

float Platform::convertRawCoordinatesToDegrees(float RawDegrees) {

float RawAsFloat = RawDegrees;

int firstdigits = ((int)RawAsFloat) / 100; // Get the first digits by turning f into an integer, then doing an integer divide by 100;

float nexttwodigits = RawAsFloat - (float)(firstdigits \* 100);

float Converted = (float)(firstdigits + nexttwodigits / 60.0);

return Converted;

}

void Platform::I2Cread(uint8\_t Address, uint8\_t Register, uint8\_t Nbytes, uint8\_t\* Data)

{

// Set register address

Wire.beginTransmission(Address);

Wire.write(Register);

Wire.endTransmission();

// Read Nbytes

Wire.requestFrom(Address, Nbytes);

uint8\_t index = 0;

while (Wire.available())

Data[index++] = Wire.read();

}

void Platform::I2CwriteByte(uint8\_t Address, uint8\_t Register, uint8\_t Data)

{

// Set register address

Wire.beginTransmission(Address);

Wire.write(Register);

Wire.write(Data);

Wire.endTransmission();

}

* + 1. Arduino.h

/\*

Arduino.h - Main include file for the Arduino SDK

Copyright (c) 2005-2013 Arduino Team. All right reserved.

This library is free software; you can redistribute it and/or

modify it under the terms of the GNU Lesser General Public

License as published by the Free Software Foundation; either

version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful,

but WITHOUT ANY WARRANTY; without even the implied warranty of

MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU

Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public

License along with this library; if not, write to the Free Software

Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

\*/

#ifndef Arduino\_h

#define Arduino\_h

#include <stdlib.h>

#include <stdbool.h>

#include <string.h>

#include <math.h>

#include <avr/pgmspace.h>

#include <avr/io.h>

#include <avr/interrupt.h>

#include "binary.h"

#ifdef \_\_cplusplus

extern "C"{

#endif

void yield(void);

#define HIGH 0x1

#define LOW 0x0

#define INPUT 0x0

#define OUTPUT 0x1

#define INPUT\_PULLUP 0x2

#define PI 3.1415926535897932384626433832795

#define HALF\_PI 1.5707963267948966192313216916398

#define TWO\_PI 6.283185307179586476925286766559

#define DEG\_TO\_RAD 0.017453292519943295769236907684886

#define RAD\_TO\_DEG 57.295779513082320876798154814105

#define EULER 2.718281828459045235360287471352

#define SERIAL 0x0

#define DISPLAY 0x1

#define LSBFIRST 0

#define MSBFIRST 1

#define CHANGE 1

#define FALLING 2

#define RISING 3

#if defined(\_\_AVR\_ATtiny24\_\_) || defined(\_\_AVR\_ATtiny44\_\_) || defined(\_\_AVR\_ATtiny84\_\_)

#define DEFAULT 0

#define EXTERNAL 1

#define INTERNAL1V1 2

#define INTERNAL INTERNAL1V1

#elif defined(\_\_AVR\_ATtiny25\_\_) || defined(\_\_AVR\_ATtiny45\_\_) || defined(\_\_AVR\_ATtiny85\_\_)

#define DEFAULT 0

#define EXTERNAL 4

#define INTERNAL1V1 8

#define INTERNAL INTERNAL1V1

#define INTERNAL2V56 9

#define INTERNAL2V56\_EXTCAP 13

#else

#if defined(\_\_AVR\_ATmega1280\_\_) || defined(\_\_AVR\_ATmega2560\_\_) || defined(\_\_AVR\_ATmega1284\_\_) || defined(\_\_AVR\_ATmega1284P\_\_) || defined(\_\_AVR\_ATmega644\_\_) || defined(\_\_AVR\_ATmega644A\_\_) || defined(\_\_AVR\_ATmega644P\_\_) || defined(\_\_AVR\_ATmega644PA\_\_)

#define INTERNAL1V1 2

#define INTERNAL2V56 3

#else

#define INTERNAL 3

#endif

#define DEFAULT 1

#define EXTERNAL 0

#endif

// undefine stdlib's abs if encountered

#ifdef abs

#undef abs

#endif

#define min(a,b) ((a)<(b)?(a):(b))

#define max(a,b) ((a)>(b)?(a):(b))

#define abs(x) ((x)>0?(x):-(x))

#define constrain(amt,low,high) ((amt)<(low)?(low):((amt)>(high)?(high):(amt)))

#define round(x) ((x)>=0?(long)((x)+0.5):(long)((x)-0.5))

#define radians(deg) ((deg)\*DEG\_TO\_RAD)

#define degrees(rad) ((rad)\*RAD\_TO\_DEG)

#define sq(x) ((x)\*(x))

#define interrupts() sei()

#define noInterrupts() cli()

#define clockCyclesPerMicrosecond() ( F\_CPU / 1000000L )

#define clockCyclesToMicroseconds(a) ( (a) / clockCyclesPerMicrosecond() )

#define microsecondsToClockCycles(a) ( (a) \* clockCyclesPerMicrosecond() )

#define lowByte(w) ((uint8\_t) ((w) & 0xff))

#define highByte(w) ((uint8\_t) ((w) >> 8))

#define bitRead(value, bit) (((value) >> (bit)) & 0x01)

#define bitSet(value, bit) ((value) |= (1UL << (bit)))

#define bitClear(value, bit) ((value) &= ~(1UL << (bit)))

#define bitWrite(value, bit, bitvalue) (bitvalue ? bitSet(value, bit) : bitClear(value, bit))

// avr-libc defines \_NOP() since 1.6.2

#ifndef \_NOP

#define \_NOP() do { \_\_asm\_\_ volatile ("nop"); } while (0)

#endif

typedef unsigned int word;

#define bit(b) (1UL << (b))

typedef bool boolean;

typedef uint8\_t byte;

void init(void);

void initVariant(void);

int atexit(void (\*func)()) \_\_attribute\_\_((weak));

void pinMode(uint8\_t, uint8\_t);

void digitalWrite(uint8\_t, uint8\_t);

int digitalRead(uint8\_t);

int analogRead(uint8\_t);

void analogReference(uint8\_t mode);

void analogWrite(uint8\_t, int);

unsigned long millis(void);

unsigned long micros(void);

void delay(unsigned long);

void delayMicroseconds(unsigned int us);

unsigned long pulseIn(uint8\_t pin, uint8\_t state, unsigned long timeout);

unsigned long pulseInLong(uint8\_t pin, uint8\_t state, unsigned long timeout);

void shiftOut(uint8\_t dataPin, uint8\_t clockPin, uint8\_t bitOrder, uint8\_t val);

uint8\_t shiftIn(uint8\_t dataPin, uint8\_t clockPin, uint8\_t bitOrder);

void attachInterrupt(uint8\_t, void (\*)(void), int mode);

void detachInterrupt(uint8\_t);

void setup(void);

void loop(void);

// Get the bit location within the hardware port of the given virtual pin.

// This comes from the pins\_\*.c file for the active board configuration.

#define analogInPinToBit(P) (P)

// On the ATmega1280, the addresses of some of the port registers are

// greater than 255, so we can't store them in uint8\_t's.

extern const uint16\_t PROGMEM port\_to\_mode\_PGM[];

extern const uint16\_t PROGMEM port\_to\_input\_PGM[];

extern const uint16\_t PROGMEM port\_to\_output\_PGM[];

extern const uint8\_t PROGMEM digital\_pin\_to\_port\_PGM[];

// extern const uint8\_t PROGMEM digital\_pin\_to\_bit\_PGM[];

extern const uint8\_t PROGMEM digital\_pin\_to\_bit\_mask\_PGM[];

extern const uint8\_t PROGMEM digital\_pin\_to\_timer\_PGM[];

// Get the bit location within the hardware port of the given virtual pin.

// This comes from the pins\_\*.c file for the active board configuration.

//

// These perform slightly better as macros compared to inline functions

//

#define digitalPinToPort(P) ( pgm\_read\_byte( digital\_pin\_to\_port\_PGM + (P) ) )

#define digitalPinToBitMask(P) ( pgm\_read\_byte( digital\_pin\_to\_bit\_mask\_PGM + (P) ) )

#define digitalPinToTimer(P) ( pgm\_read\_byte( digital\_pin\_to\_timer\_PGM + (P) ) )

#define analogInPinToBit(P) (P)

#define portOutputRegister(P) ( (volatile uint8\_t \*)( pgm\_read\_word( port\_to\_output\_PGM + (P))) )

#define portInputRegister(P) ( (volatile uint8\_t \*)( pgm\_read\_word( port\_to\_input\_PGM + (P))) )

#define portModeRegister(P) ( (volatile uint8\_t \*)( pgm\_read\_word( port\_to\_mode\_PGM + (P))) )

#define NOT\_A\_PIN 0

#define NOT\_A\_PORT 0

#define NOT\_AN\_INTERRUPT -1

#ifdef ARDUINO\_MAIN

#define PA 1

#define PB 2

#define PC 3

#define PD 4

#define PE 5

#define PF 6

#define PG 7

#define PH 8

#define PJ 10

#define PK 11

#define PL 12

#endif

#define NOT\_ON\_TIMER 0

#define TIMER0A 1

#define TIMER0B 2

#define TIMER1A 3

#define TIMER1B 4

#define TIMER1C 5

#define TIMER2 6

#define TIMER2A 7

#define TIMER2B 8

#define TIMER3A 9

#define TIMER3B 10

#define TIMER3C 11

#define TIMER4A 12

#define TIMER4B 13

#define TIMER4C 14

#define TIMER4D 15

#define TIMER5A 16

#define TIMER5B 17

#define TIMER5C 18

#ifdef \_\_cplusplus

} // extern "C"

#endif

#ifdef \_\_cplusplus

#include "WCharacter.h"

#include "WString.h"

#include "HardwareSerial.h"

#include "USBAPI.h"

#if defined(HAVE\_HWSERIAL0) && defined(HAVE\_CDCSERIAL)

#error "Targets with both UART0 and CDC serial not supported"

#endif

uint16\_t makeWord(uint16\_t w);

uint16\_t makeWord(byte h, byte l);

#define word(...) makeWord(\_\_VA\_ARGS\_\_)

unsigned long pulseIn(uint8\_t pin, uint8\_t state, unsigned long timeout = 1000000L);

unsigned long pulseInLong(uint8\_t pin, uint8\_t state, unsigned long timeout = 1000000L);

void tone(uint8\_t \_pin, unsigned int frequency, unsigned long duration = 0);

void noTone(uint8\_t \_pin);

// WMath prototypes

long random(long);

long random(long, long);

void randomSeed(unsigned long);

long map(long, long, long, long, long);

#endif

#include "pins\_arduino.h"

#endif