Ультразвуковой радар

Выполнила Дзюба Виктория, 232 группа

Основные требования

1. Финальная версия проекта должна являться законченным продуктом с полноценной документацией.
2. Код проекта должен содержать подробные комментарии, поясняющие основные моменты логики работы.
3. Styleguide не конкретизирован, однако необходимо придерживаться единого стиля во всем проекте.
4. Необходимо максимально разделить три основных компонента: обработку входных данных, вывод на дисплей и логику работы программы.
5. Необходимо полностью повторить интерфейс и геймплей оригинальной игры по ссылке, если не указано иного в примечаниях. Любые отклонения от оригинала должны быть строго аргументированы.
6. Код проекта должен быть выложен в открытом репозитории сервиса github.com.
7. Основные дедлайны:
   - 7.1. **24.09.2019**: readme с общим описанием проекта и первые предварительные наработки по логике работы системы в репозитории.
   - 7.2. **29.10.2019**: общая структура проекта + работа с периферией в репозитории.
   - 7.3. **26.11.2019**: финальная версия проекта, readme дополнен инструкцией по запуску и использованию.
   - 7.4. **03.12.2019**: презентации проектов.
8. Нарушение каждого дедлайна влечет добавление новых требований к проекту.
9. В процессе обсуждения каждого конкретного проекта требования могут быть уточнены и дополнены.

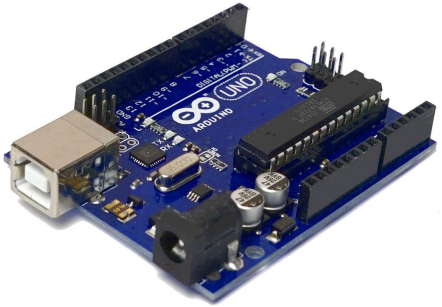
Требования к выполнению задачи

Периферия: сенсорный дисплей, 2 сонара, сервопривод
Краткое описание: 2 противоположно направленных сонара установлены на сервоприводе, поворачивающемся от 0 до 180 градусов и обратно. Реализовать визуализацию поворачивающегося луча радара и вывести точки на расстоянии, пропорциональном измеренному с сонаров.
Дополнительные задания: меню, настройки

Используемые устройства

Серводвигатель Tower Pro MG995

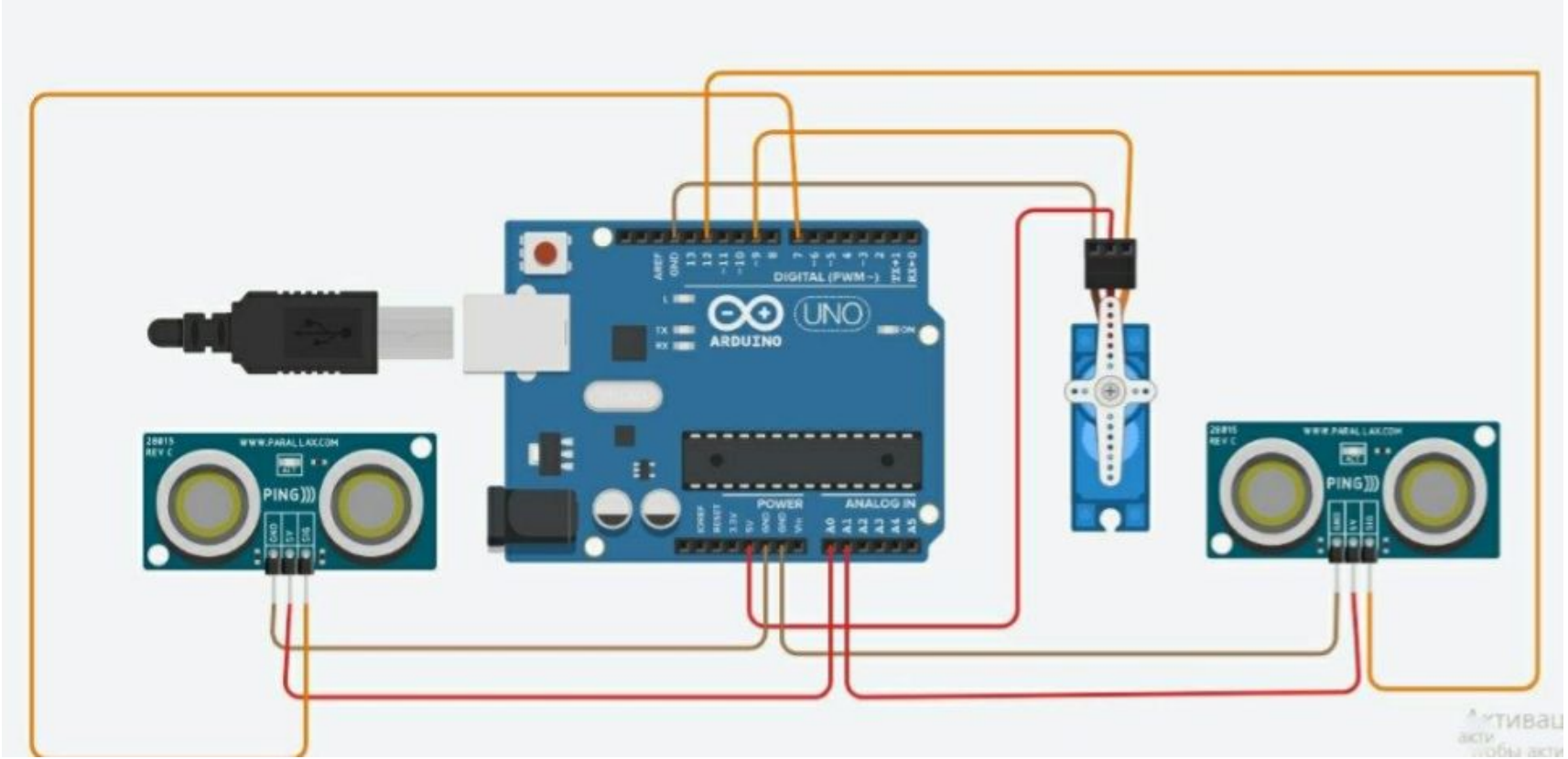
Arduino Uno



Ультразвуковой датчик HC-SR04

TFT 2.4 LCD дисплей

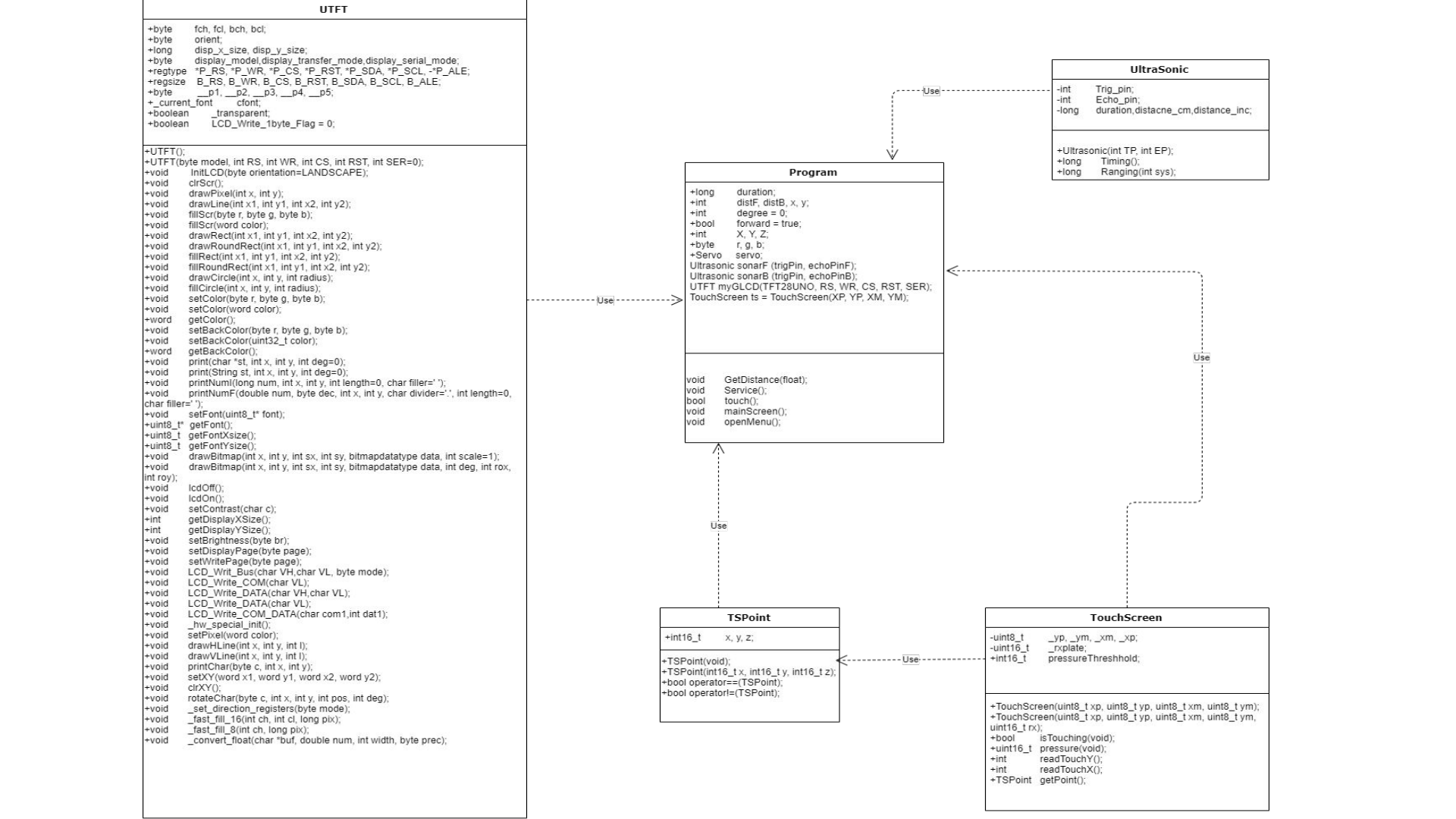
Схема подключения устройств



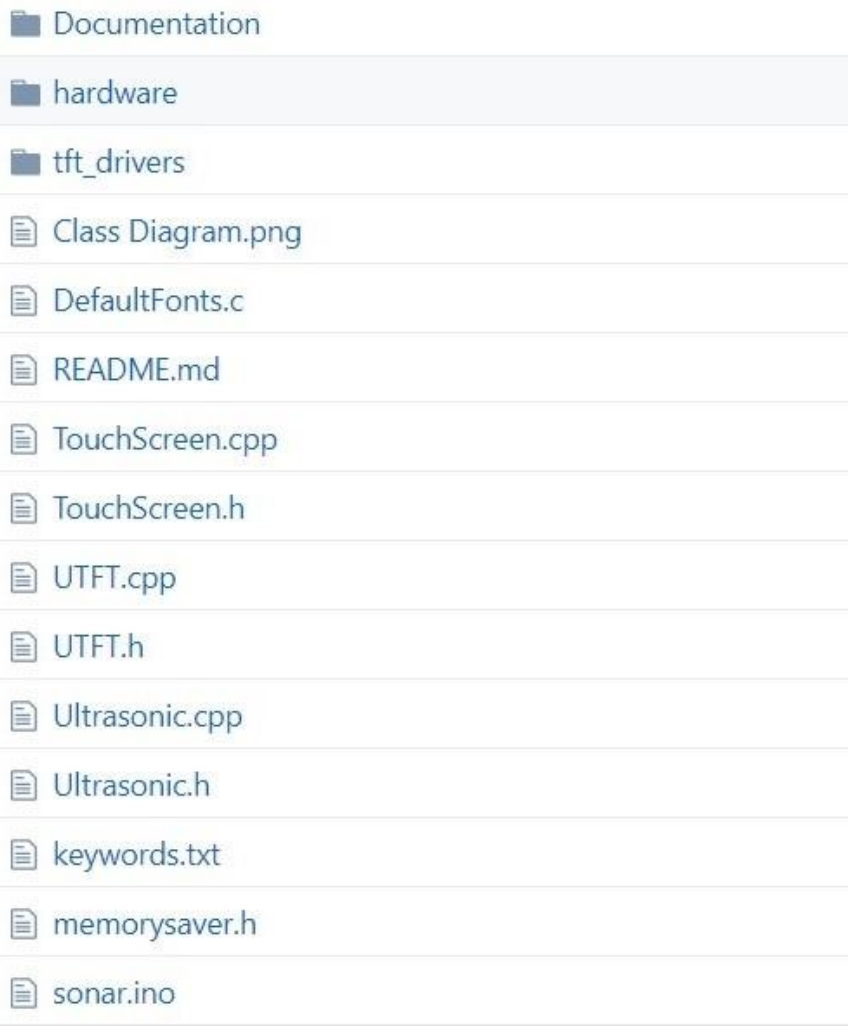
Использованные библиотеки

1)UTFT - для работы с цветным TFT дисплеем
2)TouchScreen - библиотека позволяет получать значения ,прочитанные с аналоговых выводов, которые прямо пропорциональны координатам точки касания сенсорного экрана
3)UltraSonic - для работы с ультразвуковыми датчиками
4)Servo - позволяет работать с серводвигателем
5)EEPROM - работа с энергонезависимой памятью

Программная архитектура проекта


UTFT
+byte fch, fcl, bch, bcl;
+byte orient;
+long disp_x_size, disp_y_size;
+byte display_model,display_transfer_mode,display_serial_mode;
+regtype *P_RS, *P_WR, *P_CS, *P_RST, *P_SDA, *P_SCL, -*P_ALE;
+regsize B_RS, B_WR, B_CS, B_RST, B_SDA, B_SCL, B_ALE;
+byte __p1, __p2, __p3, __p4, __p5;
+_current_font cfont;
+boolean _transparent;
+boolean LCD_Write_1byte_Flag = 0;
+UTFT();
+UTFT(byte model, int RS, int WR, int CS, int RST, int SER=0);
+void InitLCD(byte orientation=LANDSCAPE);
+void clrScr();
+void drawPixel(int x, int y);
+void drawLine(int x1, int y1, int x2, int y2);
+void fillScr(byte r, byte g, byte b);
+void fillScr(word color);
+void drawRect(int x1, int y1, int x2, int y2);
+void drawRoundRect(int x1, int y1, int x2, int y2);
+void fillRect(int x1, int y1, int x2, int y2);
+void fillRoundRect(int x1, int y1, int x2, int y2);
+void drawCircle(int x, int y, int radius);
+void fillCircle(int x, int y, int radius);
+void setColor(byte r, byte g, byte b);
+void setColor(word color);
+word getColor();
+void setBackColor(byte r, byte g, byte b);
+void setBackColor(uint32_t color);
+word getBackColor();
+void print(char *st, int x, int y, int deg=0);
+void print(String st, int x, int y, int deg=0);
+void printNumI(long num, int x, int y, int length=0, char filler=' ');
+void printNumF(double num, byte dec, int x, int y, char divider='.', int length=0, char filler=' ');
+void setFont(uint8_t* font);
+uint8_t* getFont();
+uint8_t getFontXsize();
+uint8_t getFontYsize();
+void drawBitmap(int x, int y, int sx, int sy, bitmapdatatype data, int scale=1);
+void drawBitmap(int x, int y, int sx, int sy, bitmapdatatype data, int deg, int rox, int roy);
+void lcdOff();
+void lcdOn();
+void setContrast(char c);
+int getDisplayXSize();
+int getDisplayYSize();
+void setBrightness(byte br);
+void setDisplayPage(byte page);
+void setWritePage(byte page);
+void LCD_Writ_Bus(char VH,char VL, byte mode);
+void LCD_Write_COM(char VL);
+void LCD_Write_DATA(char VH,char VL);
+void LCD_Write_DATA(char VL);
+void LCD_Write_COM_DATA(char com1,int dat1);
+void _hw_special_init();
+void setPixel(word color);
+void drawHLine(int x, int y, int l);
+void drawVLine(int x, int y, int l);
+void printChar(byte c, int x, int y);
+void setXY(word x1, word y1, word x2, word y2);
+void clrXY();
+void rotateChar(byte c, int x, int y, int pos, int deg);
+void _set_direction_registers(byte mode);
+void _fast_fill_16(int ch, int cl, long pix);
+void _fast_fill_8(int ch, long pix);
+void _convert_float(char *buf, double num, int width, byte prec);
Use
UltraSonic
-int Trig_pin;
-int Echo_pin;
-long duration,distacne_cm,distance_inc;
+Ultrasonic(int TP, int EP);
+long Timing();
+long Ranging(int sys);
Program
+long duration;
+int distF, distB, x, y;
+int degree = 0;
+bool forward = true;
+int X, Y, Z;
+byte r, g, b;
+Servo servo;
Ultrasonic sonarF (trigPin, echoPinF);
Ultrasonic sonarB (trigPin, echoPinB);
UTFT myGLCD(TFT28UNO, RS, WR, CS, RST, SER);
TouchScreen ts = TouchScreen(XP, YP, XM, YM);
void GetDistance(float);
void Service();
bool touch();
void mainScreen();
void openMenu();
Use
Use
Use
TSPoint
+int16_t x, y, z;
+TSPoint(void);
+TSPoint(int16_t x, int16_t y, int16_t z);
+bool operator==(TSPoint);
+bool operator!=(TSPoint);
TouchScreen
-uint8_t _yp, _ym, _xm, _xp;
-uint16_t _rxplate;
+int16_t pressureThreshhold;
+TouchScreen(uint8_t xp, uint8_t yp, uint8_t xm, uint8_t ym);
+TouchScreen(uint8_t xp, uint8_t yp, uint8_t xm, uint8_t ym, uint16_t rx);
+bool isTouching(void);
+uint16_t pressure(void);
+int readTouchY();
+int readTouchX();
+TSPoint getPoint();

Описание файловой архитектуры проекта



Распиновка

У LCD-shield выполнена дополнительная распиновка. Распараллелены 10-13 пины, а также пины питания и заземления.

Объект	Пин
Дисплей	
RS	A2
WR	A1
CS	A3
RST	A4
SER	A0
Радары и Серводвигатель	
TrigPin	10
EchoPinF	11
EchoPinB	12
Серводвигатель	13

Инструкция по эксплуатации

1)Сборка оборудования (пины для радара подписаны)
2)Прошивка в режиме отладки для проверки работоспособности оборудования и занесения данных в энергонезависимую память
3)Прошивка в рабочем режиме
4)Настройка скорости вращения, радиуса воздействия и цветовой палитры осуществляется через “меню”

Демонстрация

Благодарю за внимание!

Github:
@mohorka