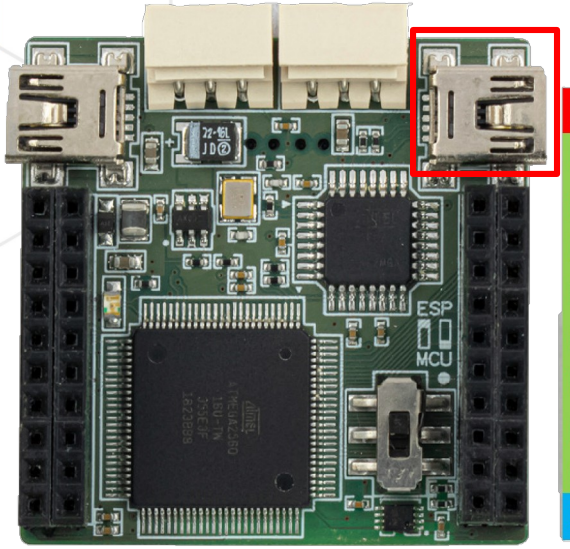
DXL IOT

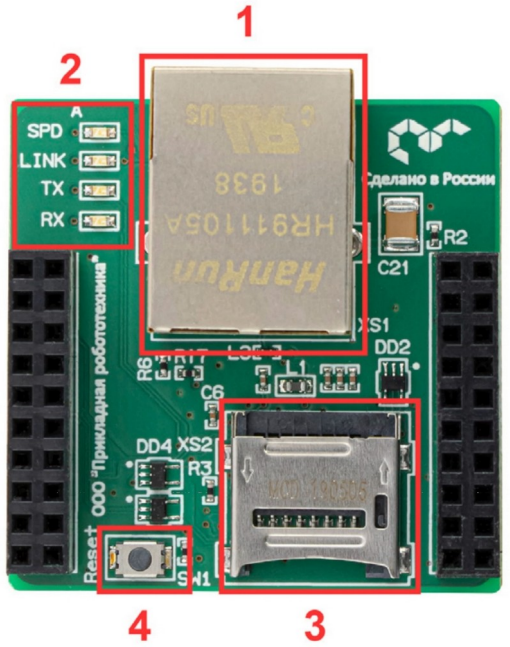
12В	3.3В
PB0 (SS)	PB1 (SCK)
PB2 (MOSI)	PB3 (MISO)
PE4 (2 - PWM)	PE5 (3 - PWM)
PG5 (4 - PWM)	PE3 (5 - PWM)
PH3 (6 - PWM)	PH4 (7 - PWM)
PH5 (8 - PWM)	PH6 (9 - PWM)
PB4 (10 - PWM)	PB5 (11 - PWM)
PB6 (13 - PWM)	PB7 (13 - PWM)
PJ0 (RXD3)	PJ1 (TXD3)



5В	GND
PF0 (A0)	PF1 (A1)
PF2 (A2)	PF3 (A3)
PF4 (A4)	PF5 (A5)
PF6 (A6)	PF7 (A7)
PK0 (A8)	PK1 (A9)
PK2 (A10)	PK3 (A11)
PK4 (A12)	PK5 (A13)
PK6 (A14)	PK7 (A15)
PD0 (SCL)	PD1 (SDA)

APPLIED ROBOTICS

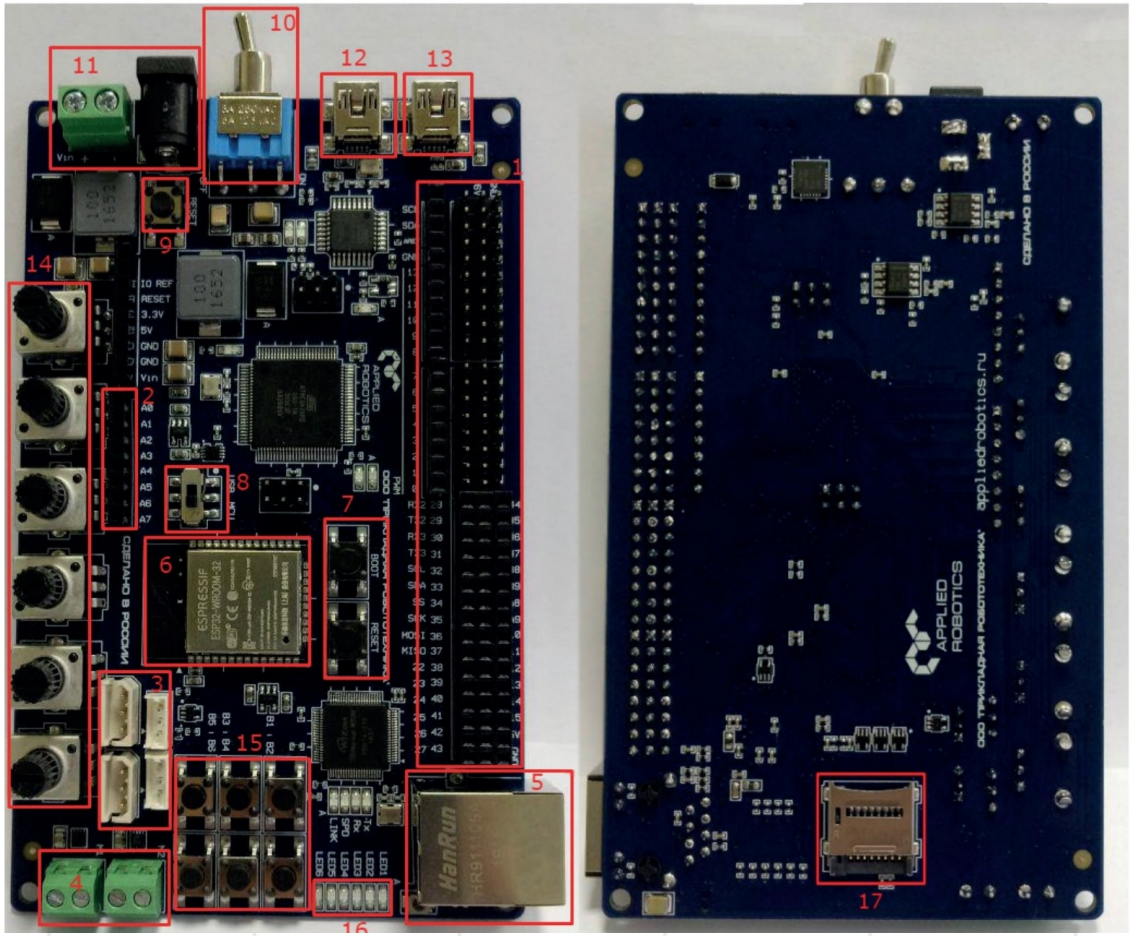
- контроллер на основе Atmega 2560 и ESP32 с возможностью функционального расширения при помощи дополнительных плат.
- программируется через среду Arduino
- имеет встроенные DXL порты



КПМИС

APPLIED ROBOTICS

- Контроллер на основе Atmega2560 и ESP32 со встроенной периферией ввода\вывода информации и сигналов.
- Встроенная периферия позволяет работать с самыми различными устройствами без необходимости установки дополнительных модулей, в то время, как DXL IOT требует расширений.
- Программируется в среде Arduino.



DXL IOT : Начало работы

APPLIED ROBOTICS

- Необходимая библиотека: DxlMaster.h

- Перед работой с устройством, необходима его инициализация:

- **DynamixelMotor motor((byte)DXL_ID)**;
- **DynamixelDevice device((byte)DXL_ID)**;

- В setup():

- **DxlMaster.begin(BAUD)**; 57600 для периферийных устройств.
- **motor.init()**;
- **motor.wheelMode()**;

- **device.init()**;

DXL IOT : Управляющие команды

APPLIED ROBOTICS

Для сервоприводов :

- void **led**(uint8_t aState); вкл\выкл LED

- void **jointMode(uint16_t aCWLimit=0, uint16_t aCCWLimit=0x3FF)**; Включает режим сервопривода
- void **enableTorque(bool aTorque=true)**; Включает нагрузки сервопривода
- void **goalPosition(uint16_t aPosition)**; Задаёт целевую позицию сервопривода, в которую он начинает стремиться.
- uint16_t **currentPosition();** Возвращает позицию сервопривода.

- void **wheelMode()**; Включает режим мотора
- void **speed(int16_t aSpeed)**; Задаёт скорость вращения

DXL IOT : Управляющие команды

APPLIED ROBOTICS

Для датчиков:

- **device.write(REG, value)**; записывает состояние на регистр устройства(напр. включает\выключает светодиод).
- **device.read(REG, data)**; Считывает с регистра данные в заранее обьявленную переменную data.

DXL IOT : Датчики

APPLIED ROBOTICS

Название, ID	Имя регистра	Адрес	Тип	Диапазон	Чтение / Запись	Описание
Модуль датчика линии ID 10	LINE_DATA	**27**	uint8_t	0..1	R	Возвращает состояние датчика линии
Модуль концевого микропереключателя ID 23	USW_DATA	**27**	uint8_t	0..1	R	Возвращает состояние микропереключателя
Модуль трехцветного светодиода ID 21	GREEN_LED_DATA	**26**	uint8_t	0..255	W	Задает интенсивность свечения зеленой составляющей светодиода
	RED_LED_DATA	**27**	uint8_t	0..255	W	Задает интенсивность свечения красной составляющей светодиода
	BLUE_LED_DATA	**28**	uint8_t	0..255	W	Задает интенсивность свечения синей составляющей светодиода
Модуль тактовой кнопки ID 3	BUT_DATA	**27**	uint8_t	0..1	R	Возвращает состояние кнопки

DXL IOT : Датчики. Примеры работы.

Модуль датчика линии ID 10	LINE_DATA	**27**	uint8_t	0..1	R	Возвращает состояние датчика линии

APPLIED ROBOTICS

- **До setup()**:

```
#include “DxlMaster.h”
DynamixelDevice device(10);
```

setup():

```
DxlMaster.init(57600);
device.init();
```

Далее там же в **setup()** либо **loop()**:

```
uint8_t val;
device.read(27, val); //в val записываются данные с датчика.
```

DXL IOT : Датчики. Примеры работы.

Модуль трехцветного светодиода ID 21	GREEN_LED_DATA	**26**	uint8_t	0..255	W	Яркость зелёного
	RED_LED_DATA	**27**	uint8_t	0..255	W	Яркость красного
	BLUE_LED_DATA	**28**	uint8_t	0..255	W	Яркость синего

- **До setup()**:

#include "DxlMaster.h"

DynamixelDevice device(21);

setup():

DxlMaster.init(57600);

device.init();

Далее там же в **setup()** либо **loop()**:

device.write(26, 255); //включаем зелёный.

device.write(27, 255); //включаем красный.

device.write(26, 255); //включаем синий.

В итоге изза смешения цветов будет гореть белый цвет.

UDP Interface

APPLIED ROBOTICS

Необходимы библиотеки
<Ethernet.h>
<EthernetUdp.h>
IPAddress MyIP(192, 168, 42, 1); //IP Адреса задаются и обрабатываются через сущность IPAddress.

До **setup()**:

```
#include <Ethernet.h>
#include <EthernetUdp.h>
#define UDP_PORT 8888
IPAddress MyIP(192, 168, 42, 25); //Статический адрес устройства
EthernetUDP udp; //Объявление сущности UDP соединения
char packetBuffer(UDP_TX_PACKET_MAX_SIZE); //Буфер обмена пакетами UDP
byte mac(6); //Массив с MAC-адрессом, будет генерироваться из MyIP
```

UDP Interface

APPLIED ROBOTICS

В **setup()**:

```
mac(0) = MyIP(0);mac(1) = MyIP(1);mac(2) = MyIP(2);
mac(3) = MyIP(3);mac(4) = MyIP(2);mac(5) = MyIP(3); //Генерация MAC адреса
Ethernet.begin(mac, MyIP); //Инициализируем Ethernet соединение.
udp.begin(UDP_PORT); //Открываем порт UDP 8888
```

В **loop()**:

```
if(udp.parsePacket()){//Если в порт UDP получен пакет
IPAddress remote = udp.remoteIP(); // Определение IP отправителя пакета
udp.read(packetBuffer, UDP_TX_PACKET_MAX_SIZE);//Считываем в массив весь пакет
//...производим обработку данных пакета…
char() ans = `answer’;
udp.beginPacket(remote, UDP_PORT); //Начинаем передачу ответа
udp.write(ans); //Отправляем пакет с ответом
udp.endPacket(); //Завершаем передачу
}
```

Пульт управления

APPLIED ROBOTICS

- Реагирует на UDP пакет формата:
  r:1:1:1:1:0:Hello#
  r: (1|0) :(1|0): (1|0) : (1|0): (0-3) : (String) #
  r:красный:синий:зелёный:жёлтый:номер_строки:сообщение#

- Постоянно присылает серверу статус формата:
  R:19:0:0:0:0#
  R:IP: (1|0) : (INT) : (INT) : (INT) #
  R:IP:Стоп_кнопка:Зел_кнопка:Жёлт_кнопка:Крас_кнопка#

- Есть возможность изменения настроек через Serial соединение в среде arduino.

Смарт-лампа

APPLIED ROBOTICS

- Реагирует на UDP пакет формата:
  I:1:1:1:1#
  I: (0|1) : (0|1) : (0|1) : (0|1) #
  I:Красный:Синий:Зелёный:Жёлтый#

- Есть возможность настройки через Serial соединение в среде arduino.