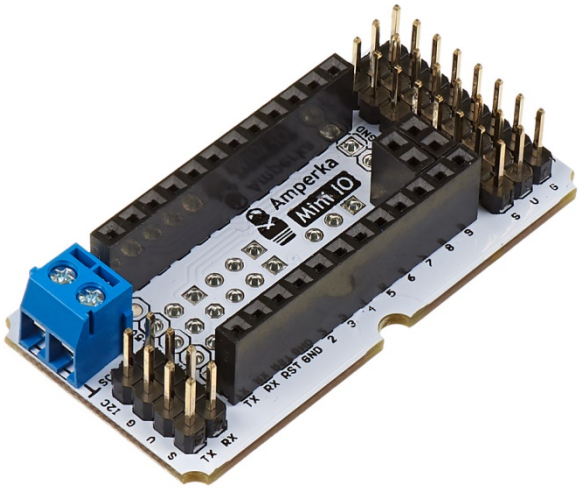
Troyka Mini IO

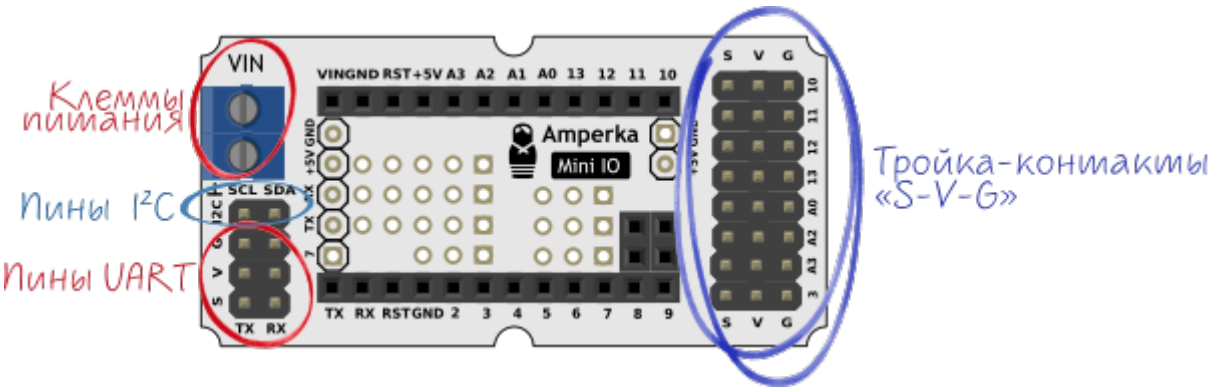
Troyka Mini IO — это двухюнитовая плата расширения, которая позволяет подключать к микроконтроллерам формата Arduino Mini большое количество модулей и сенсоров через стандартные трёхпроводные шлейфы. Это позволяет не прибегать к пайке или отдельной макетной плате.



Подключение

Mini IO предназначена для работы с платами форм-фактора Arduino Mini. Установите микроконтроллер на плату расширения в соответствии с маркировкой пинов.

Элементы платы



Тройка-контакты «S-V-G»

Контакты для подключения модулей и сенсоров соединены с линиями управляющей платы следующим образом:

- сигнал (S) — с соответствующим цифровым или аналоговым пином;
- питание (V) — с рабочим напряжением;
- земля (G) — с землёй.

Для подключения доступны:

- цифровые пины `3`, `10`, `11`, `12` и `13`;

- аналоговые пины `A0`, `A2` и `A3`.

  Из за технического стандарта форм-фактора Troyka-модулей на плате не распаяны пины `A1` и `4`.

  Контакты интерфейса I²C

  Контакты для подключения устройств, которые общаются с управляющей электроникой по протоколу I²C / TWI.

  Контакты интерфейса UART

  Контакты для подключения устройств, поддерживающими последовательный интерфейс обмена данными. Эти же пины используются для прошивки контроллера.

  Клеммы питания

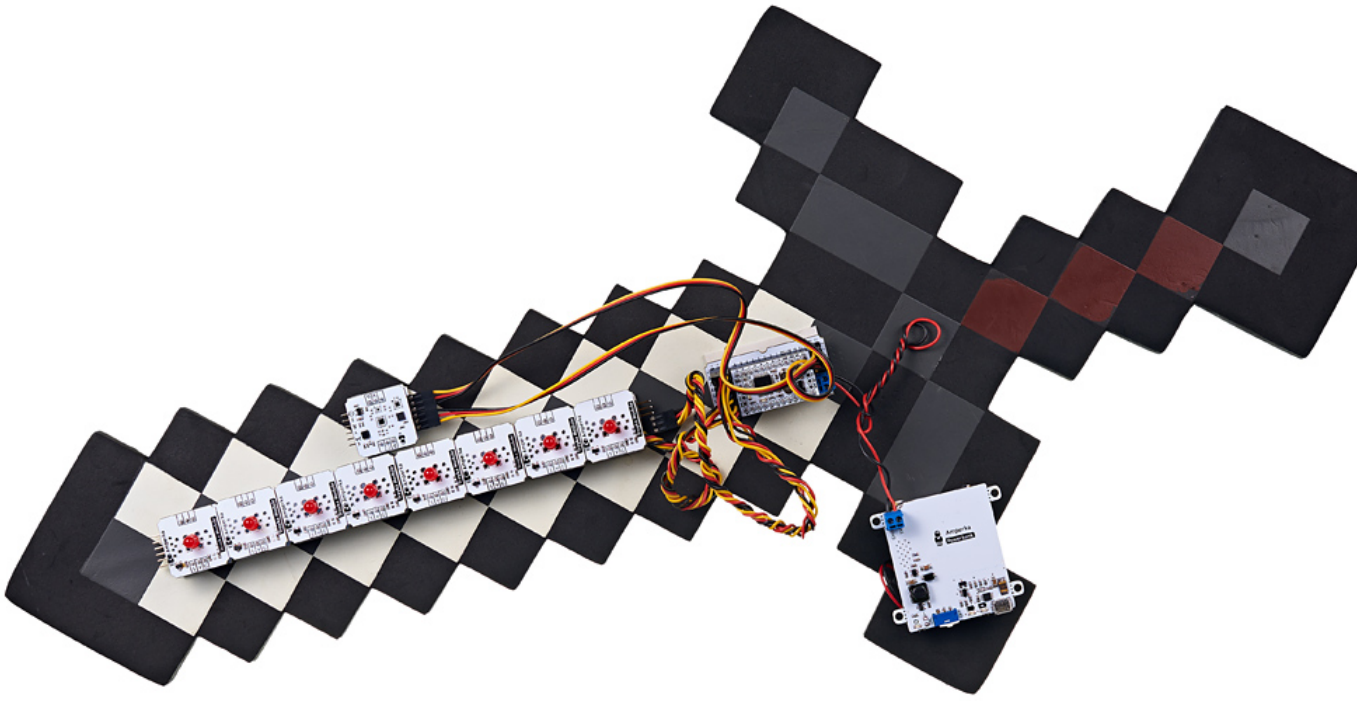
  Для подключения источника питания служат две винтовые клеммы. Подключите от 5,3 В до 9 В к клемме с пометкой `VIN`. Минусовая клемма обозначена знаком `⍰`.

  Монтажная площадка

  В центре модуля предусмотрены лужённые отверстия для самостоятельной распайки пинов `2` и с `4` по `9`. Отдельно выделены контакты для распайки аппаратного интерфейса ICSP.

POV-меч

Чтобы продемонстрировать работу модуля, мы собрали POV строку из 8 светодиодов — на Troyka-модулях и без макетки.



Светодиоды мы подключили к Troyka-контактам, а акселерометр — к контактам `I²C` для обмена данными одним проводом и вторым к одной из свободных троек `UART`. Для крепления модулей мы использовали одно- и четырёхюнитовые Pad-ы. Источником питания стал Power Bank.

Исходный код

povdisplay.ino

```
// библиотека для работы I²C
#include <Wire.h>

// библиотека для работы с модулями IMU
#include <TroykaIMU.h>

// задержка между столбцами одного символа
// ширина символа
#define TIMER_COLUMS 3

// задержка между символами
// ширина между символами
#define TIMER_FRAMES 8

// количество светодиодов, высота символа
#define ROW  8
// количество столбцов в символе
#define COL  5

// массив пинов, к которым подключены светодиоды
int pins[] = {3, A3, A2, A0, 13, 12, 11, 10};

// создаём объект для работы с акселерометром
Accelerometer accel;

// переменная для хранения
// направления и величины ускорения по оси Y
float y = 0;

// переменная для хранения кол-во раз
// подряд положительных значений направленного ускорения
int y_pol = 0;

// переменная для хранения кол-во раз
// подряд отрицательных значений направленного ускорения
int y_neg = 0;

// двумерный массив символов
const unsigned char image[][COL] = {
  {0x00, 0x00, 0x00, 0x00, 0x00},  // space
  {0x00, 0xF6, 0xF6, 0x00, 0x00},  // !
  {0x00, 0xE0, 0x00, 0xE0, 0x00},  // "
```

```
{0x28, 0xFE, 0x28, 0xFE, 0x28},  // #
{0x00, 0x64, 0xD6, 0x54, 0x08},  // $
{0xC2, 0xCC, 0x10, 0x26, 0xC6},  // %
{0x4C, 0xB2, 0x92, 0x6C, 0x0A},  // &
{0x00, 0x00, 0xE0, 0x00, 0x00},  // '
{0x00, 0x38, 0x44, 0x38, 0x00},  // )
{0x00, 0x82, 0x44, 0x38, 0x00},  // )
{0x88, 0x50, 0xF8, 0x50, 0x88},  // *
{0x08, 0x08, 0x3E, 0x08, 0x08},  // +
{0x00, 0x00, 0x05, 0x06, 0x00},  // ,
{0x08, 0x08, 0x08, 0x08, 0x08},  // -
{0x00, 0x00, 0x06, 0x06, 0x00},  // .
{0x02, 0x0C, 0x10, 0x60, 0x80},  // /
{0x7C, 0x8A, 0x92, 0xA2, 0x7C},  // 0
{0x00, 0x42, 0xFE, 0x02, 0x00},  // 1
{0x42, 0x86, 0x8A, 0x92, 0x62},  // 2
{0x44, 0x82, 0x92, 0x92, 0x6C},  // 3
{0x10, 0x30, 0x50, 0xFE, 0x10},  // 4
{0xE4, 0xA2, 0xA2, 0xA2, 0x9C},  // 5
{0x3C, 0x52, 0x92, 0x92, 0x0C},  // 6
{0x80, 0x86, 0x98, 0xE0, 0x80},  // 7
{0x6C, 0x92, 0x92, 0x92, 0x6C},  // 8
{0x60, 0x92, 0x92, 0x94, 0x78},  // 9
{0x00, 0x00, 0x36, 0x36, 0x00},  // :
{0x00, 0x00, 0x35, 0x36, 0x00},  // ;
{0x10, 0x28, 0x44, 0x82, 0x00},  // <
{0x28, 0x28, 0x28, 0x28, 0x28},  // =
{0x00, 0x82, 0x44, 0x28, 0x10},  // >
{0x40, 0x80, 0x8A, 0x90, 0x60},  // ?
{0x7C, 0x82, 0xBA, 0xBA, 0x62},  // @
{0x3E, 0x48, 0x88, 0x48, 0x3E},  // A
{0xFE, 0x92, 0x92, 0x92, 0x6C},  // B
{0x7C, 0x82, 0x82, 0x82, 0x44},  // C
{0xFE, 0x82, 0x82, 0x82, 0x7C},  // D
{0xFE, 0x92, 0x92, 0x92, 0x82},  // E
{0xFE, 0x90, 0x90, 0x90, 0x80},  // F
{0x7C, 0x82, 0x82, 0x8A, 0x4E},  // G
{0xFE, 0x10, 0x10, 0x10, 0xFE},  // H
{0x82, 0x82, 0xFE, 0x82, 0x82},  // I
{0x84, 0x82, 0xFC, 0x80, 0x80},  // J
{0xFE, 0x10, 0x28, 0x44, 0x82},  // K
{0xFE, 0x02, 0x02, 0x02, 0x02},  // L
{0xFE, 0x40, 0x20, 0x40, 0xFE},  // M
{0xFE, 0x60, 0x10, 0x0C, 0xFE},  // N
{0x7C, 0x82, 0x82, 0x82, 0x7C},  // O
{0xFE, 0x90, 0x90, 0x90, 0x60},  // P
{0x7C, 0x82, 0x82, 0x86, 0x7E},  // Q
{0xFE, 0x90, 0x98, 0x94, 0x62},  // R
{0x64, 0x92, 0x92, 0x92, 0x4C},  // S
{0x80, 0x80, 0xFE, 0x80, 0x80},  // T
```

```
  {0xFC, 0x02, 0x02, 0x02, 0xFC},  // U
  {0xF8, 0x04, 0x02, 0x04, 0xF8},  // V
  {0xFC, 0x02, 0x0C, 0x02, 0xFC},  // W
  {0xC6, 0x28, 0x10, 0x28, 0xC6},  // X
  {0xC0, 0x20, 0x1E, 0x20, 0xC0},  // Y
  {0x86, 0x8A, 0x92, 0xA2, 0xC2},  // Z
  {0x00, 0x00, 0xFE, 0x82, 0x00},  // [
  {0x00, 0x00, 0x00, 0x00, 0x00},  // \
  {0x80, 0x60, 0x10, 0x0C, 0x02},  // ]
  {0x20, 0x40, 0x80, 0x40, 0x20},  // ^
  {0x01, 0x01, 0x01, 0x01, 0x01},  // _
  {0x80, 0x40, 0x20, 0x00, 0x00},  // `
  {0x04, 0x2A, 0x2A, 0x2A, 0x1E},  // a
  {0xFE, 0x12, 0x22, 0x22, 0x1C},  // b
  {0x1C, 0x22, 0x22, 0x22, 0x14},  // c
  {0x1C, 0x22, 0x22, 0x12, 0xFE},  // d
  {0x1C, 0x2A, 0x2A, 0x2A, 0x18},  // e
  {0x10, 0x7E, 0x90, 0x80, 0x40},  // f
  {0x18, 0x25, 0x25, 0x25, 0x1E},  // g
  {0xFE, 0x10, 0x10, 0x10, 0x0E},  // h
  {0x00, 0x12, 0x5E, 0x02, 0x00},  // i
  {0x02, 0x01, 0x01, 0x11, 0x5E},  // j
  {0xFE, 0x08, 0x08, 0x14, 0x22},  // k
  {0x00, 0x82, 0xFE, 0x02, 0x00},  // l
  {0x3E, 0x20, 0x1C, 0x20, 0x1E},  // m
  {0x3E, 0x20, 0x20, 0x20, 0x1E},  // n
  {0x1C, 0x22, 0x22, 0x22, 0x1C},  // o
  {0x3F, 0x24, 0x24, 0x24, 0x18},  // p
  {0x18, 0x24, 0x24, 0x3F, 0x01},  // q
  {0x3E, 0x10, 0x20, 0x20, 0x10},  // r
  {0x12, 0x2A, 0x2A, 0x2A, 0x04},  // s
  {0x00, 0x10, 0x3C, 0x12, 0x04},  // t
  {0x3C, 0x02, 0x02, 0x02, 0x3E},  // u
  {0x30, 0x0C, 0x02, 0x0C, 0x30},  // v
  {0x38, 0x06, 0x18, 0x06, 0x38},  // w
  {0x22, 0x14, 0x08, 0x14, 0x22},  // x
  {0x38, 0x05, 0x05, 0x05, 0x3E},  // y
  {0x22, 0x26, 0x2A, 0x32, 0x22},  // z
  {0x00, 0x10, 0x6C, 0x82, 0x82},  // {
  {0x00, 0x00, 0xFF, 0x00, 0x00},  // |
  {0x82, 0x82, 0x6C, 0x10, 0x00},  // }
  {0x08, 0x10, 0x18, 0x08, 0x10},  // ~
  {0x38, 0x7c, 0x3e, 0x7c, 0x38},  // Love 127
};

void setup()
{
  // инициализация акселерометра
  accel.begin();
  // устанавливаем чувствительность
```

```
  accel.setRange(RANGE_2G);
  // устанавливаем пины из массива pins[] в режим выхода
  for (int i = 0; i < ROW; i++) {
    pinMode(pins[i], OUTPUT);
  }
  // гасим все светодиоды
  ledsAllOff();
}

void loop()
{
  // считываем направления и величину ускорения по оси Y
  y = accel.readAY());
  // если ускорение превысило допустимое значение
  // и направлено с права налево
  if (y > 15) {
    y_pol++;
    if (y_pol == 60) {
      y_pol = 0;
      // так как ускорение направлено справа налево
      // выводим буквы в обратном порядке
      // функцией которая их зеркально отражает
      showRL('o');
      showRL('a');
      showRL('i');
      showRL('C');
      delay(10);
    }
    // если ускорение превысило допустимое значение
    // и направлено с лево на право
  } else if (y < - 15) {
    y_neg++;
    if (y_neg == 60) {
      y_neg = 0;
      // так как ускорение направлено слева направо
      // выводим буквы в обычном порядке
      showLR('C');
      showLR('i');
      showLR('a');
      showLR('o');
      delay(10);
    }
  }
}

// функция вывода символа слева направо
void showLR(unsigned char c)
{
  // перебор столбцов матрицы
  for (int i = 0; i < COL; i++) {
```

```
    // перебор строк матрицы (светодиодов)
    ledsSet(c, i);
    // задержка задающая ширину символа
    delay(TIMER_COLUMS);
  }

  // гасим все светодиоды
  ledsAllOff();
  // задержка задающая ширину между символами
  delay(TIMER_FRAMES);
}

// функция вывода символа справа налево
void showRL(unsigned char c)
{
  // перебор столбцов матрицы
  for (int i = COL - 1; i >= 0; i--) {
    // перебор строк матрицы (светодиодов)
    ledsSet(c, i);
    // задержка задающая ширину символа
    delay(TIMER_COLUMS);
  }
  // гасим все светодиоды
  ledsAllOff();
  // задержка задающая ширину между символами
  delay(TIMER_FRAMES);
}

// функция гасящая все светодиоды
void ledsAllOff()
{
  for (int i = 0; i < ROW; i++) {
    digitalWrite(pins[i], LOW);
  }
}

// функция включения светодиодов в текущем столбце
void ledsSet(unsigned char c, int i)
{
  for (int j = 0; j < ROW; j++) {
    digitalWrite(pins[j], bitRead(image[c-32][i], j));
  }
}
```

Ваш светодиодный меч готов!