Overview

The FA-DUINO is an Arduino-based industrial controller. It has features similar to those found in other Arduino products and can be programmed with the Arduino IDE. Other Arduino products are not very well suited for fields such as factory automation. The FA-DUINO has been designed to handle signals of higher voltage and current, and remove the burden of external circuit design and fabrication from the user.

The FA-DUINO requires only simple connections to its terminal blocks and headers without the need for external peripherals.

The FA-DUINO-24RA

The FA-DUINO-24RA has a built-in Mega2560 MCU

- Program memory: 256KB
- SRAM: 8KB
- EEPROM: 4KB
- Clock Speed: 16MHz
- 16 - 24VDC Inputs (pins 30~45)
- 8 - 10A Relay Outputs (pins 22~29)
- 2 - RS-232C Communication Port
- 1 - RS-485 Communication Port
- 4 - 0~10V Analog Inputs
- 4 - 0~20mA Analog Inputs
- 10-bit ADC (0~1023)
- Powered by 24VDC
- Operating Temperature: 0 ~ 60℃
- Operating Humidity: 35 ~ 85% RH

Programming the FA-DUINO

The FA-DUINO can be programmed using the Arduino IDE available from [http://arduino.cc](http://arduino.cc). Select the Arduino Mega 2560 option from the Tools-->Board menu. Select the PC COM port that the FA-DUINO is connected to from the Tools-->Serial Port menu. You can find the COM port number in Windows Device Manager.

Write your program, and click the “Upload” icon to compile the program and upload to the FA-DUINO.
### IO Map

<table>
<thead>
<tr>
<th>Direction</th>
<th>Pins</th>
<th>Input Voltage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>30–45</td>
<td>0 or 24VDC (20–28V is logic high)</td>
<td>24V = Logic High 0V = Logic Low</td>
</tr>
<tr>
<td>Output</td>
<td>22–29</td>
<td>1A Relay Outputs</td>
<td>Logic High = On Logic Low = Off</td>
</tr>
<tr>
<td>Analog Input</td>
<td>0–7</td>
<td>0–3 = 0–20mA 4–7 = 0–10V</td>
<td>VA = analogRead(A0) // Read channel 0 VA = analogRead(A4) // Read channel 4</td>
</tr>
</tbody>
</table>

**Example:**

```c
int val = 0; // Digital input variable
int AD_val = 0; // Read ADC value
pinMode(22, OUTPUT); // Set pin 22 to output
val = digitalRead(30); // Read from digital input 30
digitalWrite(22, val); // Set digital output to same value as digital input
AD_val = analogRead(A0); // Read ADC channel 0
```

### Status LED

The FA-DUINO has a programmable status LED on pin 13 for providing visual indication to the operator.

**Example:**

```c
digitalWrite(13, HIGH); // Turn status LED on
delay(1000); // Delay for 1 second
digitalWrite(13, LOW); // Turn status LED off
delay(1000); // Delay for 1 second
```

### Connecting the Upload Cable

**Physical Connections**

- **Mounting feet**
- **Screw terminal block for input**
- **Led display**
- **Din Rail Mounting holder**
- **Download port**
- **RS232C channel 1 port**
- **0–20mA Analog input port**
- **0–10VDC Analog input port**

**Diagram:**

- **PC (9PIN MALE) to FA-DUINO 24RA**
  - **TX**, **RX**, **GND**
  - **5V**, **3.3V**, **DTR**
  - **USB to RS232**
  - **MODBUS MASTER**
  - **RS485 Device 24V**
  - **RS232 Device**
Using Proximity Sensors
Proximity sensors can be used to detect the existence, movement, and displacement of objects without any physical contact with the object. They are used quite often in the field of automation.

DC 2-Wire Model
Sensor output connected in reverse

DC 3-Wire Model (PNP type)
Sensor output connected in reverse

DC 3-Wire Model (NPN type)
Sensor output connected in reverse

Digital I/O Specifications

<table>
<thead>
<tr>
<th>Specifications</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Inputs</td>
<td>16</td>
</tr>
<tr>
<td>Input Voltage Range</td>
<td>20VDC ~ 28VDC</td>
</tr>
<tr>
<td>Recommended Operating Voltage</td>
<td>24VDC</td>
</tr>
<tr>
<td>On/Off Switching Speed</td>
<td>10ms (Ladder Scan Time is 10ms)</td>
</tr>
<tr>
<td>Input Impedance</td>
<td>2.2kΩ @ 24VDC</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output Relay Specifications</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Outputs</td>
<td>8</td>
</tr>
<tr>
<td>Input Voltage Range</td>
<td>5 ~ 30VDC / 4 ~ 264VAC</td>
</tr>
<tr>
<td>Recommended Operating Voltage</td>
<td>6 ~ 27VDC / 6 ~ 240VAC</td>
</tr>
<tr>
<td>On/Off frequency</td>
<td>10Hz (10 times per second)</td>
</tr>
<tr>
<td>Maximum Current</td>
<td>10A per relay</td>
</tr>
<tr>
<td>Minimum Current</td>
<td>100mA per relay</td>
</tr>
</tbody>
</table>

Analog I/O Specifications

<table>
<thead>
<tr>
<th>Analog Current Input (0 ~ 3) Specification</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolution and Error</td>
<td>10-bit, +/- 2%</td>
</tr>
<tr>
<td>Input Current Range</td>
<td>0mA ~ 22mA</td>
</tr>
<tr>
<td>Recommended Operating Current</td>
<td>4mA ~ 20mA</td>
</tr>
<tr>
<td>Type</td>
<td>Non-isolated, Built-in LPF</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Analog Voltage Input (4 ~ 7) Specifications</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolution and Error</td>
<td>10-bit, +/- 2%</td>
</tr>
<tr>
<td>Input Voltage Range</td>
<td>-0.5VDC ~ 10.5VDC</td>
</tr>
<tr>
<td>Operating Voltage</td>
<td>0VDC ~ 10VDC</td>
</tr>
<tr>
<td>Type</td>
<td>Non-isolated, Built-in LPF</td>
</tr>
</tbody>
</table>

Communication Specifications

<table>
<thead>
<tr>
<th>Communication Port Specifications</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>RS-232 (+/- 10VDC)</td>
</tr>
<tr>
<td>Flow Control</td>
<td>No RTS Flow Control</td>
</tr>
<tr>
<td>Maximum Baud Rate</td>
<td>115200</td>
</tr>
<tr>
<td>Maximum Distance</td>
<td>2 meters</td>
</tr>
</tbody>
</table>
**Simple Examples**

**Example 1 – Flashing the Status LED**
```c
const int StatusLED = 13;
void setup()
{
  pinMode(StatusLED, OUTPUT);
}
void loop()
{
  digitalWrite(StatusLED, HIGH);
  delay(1000);
  digitalWrite(StatusLED, LOW);
  delay(1000);
}
```

**Example 2 – Toggling a Relay**
```c
const int Relay22 = 22;
void setup()
{
  pinMode(Relay22, OUTPUT);
}
void loop()
{
  digitalWrite(Relay22, HIGH);
  delay(1000);
  digitalWrite(Relay22, LOW);
  delay(1000);
}
```

**Example 3 – Input and Output Control**
```c
const int StatusLED = 13;
const int Relay22 = 22;
const int Input_30 = 4;
void setup()
{
  pinMode(StatusLED, OUTPUT);
  pinMode(Relay22, OUTPUT);
  pinMode(Input_30, INPUT);
}
void loop()
{
  if(HIGH==digitalRead(Input_30))
  {
    digitalWrite(StatusLED, HIGH);
    digitalWrite(Relay22, HIGH);
  }
  else
  {
    digitalWrite(StatusLED, LOW);
    digitalWrite(Relay22, LOW);
  }
}
```

**Example 4 – Analog Input with the Serial Monitor**
```c
int ADI_Value0;
int ADV_Value4;
void setup()
{
  Serial.begin(9600);
}
void loop()
{
  ADI_Value0 = analogRead(A0);
  delay(100);
  ADV_Value4 = analogRead(A4);
  delay(100);
  Serial.print(" CH 0 = ");
  Serial.print(ADI_Value0);
  Serial.print("  CH 4 = ");
  Serial.print(ADV_Value4);
  Serial.println("\n");
  delay(200);
}
```

**Interfacing with the UIF-5K**

The UIF-5K is a 5-key character LCD panel that can be used in conjunction with the FADUINO to add a simple user interface.

**Description**

1. LED Indicator
2. Character LCD (Characters, Numbers and Symbols)
3. Key Value for RS-232 Communication:
   - F1 – 0x01 (1 byte)
   - F2 – 0x02 (1 byte)
   - F3 – 0x03 (1 byte)
   - F4 – 0x04 (1 byte)
   - F5 – 0x05 (1 byte)

The following shows how to connect the two together. The UIF-5K must be powered separately with a 9V-24V supply.
Example 1
The following source code will output text to the UIF-5K’s display.

```cpp
void setup()
{
  Serial1.begin(115200);   // Baud rate 115200
  uif_clear();             // Clear the display
  delay(20);               // System delay
  uif_buzzer(1);           // Buzzer on
  delay(20);               // System delay
  uif_locate(0,0);         // Set the cursor to the given x & y coordinates
  Serial1.print("=== UIF 5K_TEST ===");
  delay(100);  uif_locate(2,1);
  Serial1.print(" FA-DUINO-24RA ");
  delay(100);
}

void loop()
{
  // Clear the display
  void uif_clear();
  Serial1.write(0x1b);
  Serial1.write(0x43);
  // Set the cursor to the given x & y coordinates
  void uif_locate(unsigned char x, unsigned char y)
  {
    Serial1.write(0x1b);
    Serial1.write(0x4C);
    Serial1.write(x);
    Serial1.write(y);
  }
  // Turn the buzzer on (1) or off (0)
  void uif_buzzer(unsigned char on_off)
  {
    Serial1.write(0x1b);
    Serial1.write(0x5a);
    Serial1.write(on_off);
  }
  // Clear the display
  void uif_clear();
}
```

Example 2
The following example will display the result of a button press on the UIF-5K’s display.

```cpp
int cnt = 0;
void setup()
{
  Serial1.begin(115200);     // baud rate 115200
  uif_clear();               // Clear the display
  delay(20);
  uif_light(1); delay(20);  // backlight on
  uif_buzzer(1); delay(20);  // buzzer on
  delay(100);  uif_locate(0,0);
  Serial1.print("=== UIF 5K_TEST ===");
  delay(100);  uif_locate(2,1);
  Serial1.print("comfiletech.com");
  delay(100);  uif_locate(2,2);
  Serial1.print("COUNTER : ");
  delay(100);  uif_locate(2,3);
  Serial1.print("BUTTON :");
  delay(100);
}

void loop()
{
  cnt++;                   // Increment the counter
  void uif_clear();
  Serial1.write(0x1b);
  Serial1.write(0x43);
  // Set the cursor to the given x & y coordinates
  void uif_locate(unsigned char x, unsigned char y)
  {
    Serial1.write(0x1b);
    Serial1.write(0x4C);
    Serial1.write(x);
    Serial1.write(y);
  }
  // Turn the buzzer on (1) or off (0)
  void uif_buzzer(unsigned char on_off)
  {
    Serial1.write(0x1b);
    Serial1.write(0x5a);
    Serial1.write(on_off);
  }
  // Clear the display
  void uif_clear();
}
```

```cpp
void serial1Event()
{
  // Display the value of the button pressed
  while (Serial1.available())
  {
    char inChar = (char)Serial1.read();
    if (inChar >= '0' && inChar <= '9')
    {
      Serial1.print(inChar, DEC);
    }
  }
  void uif_clear()
  {
    Serial1.write(0x1b);
    Serial1.write(0x43);
  }
  // Turn the backlight on (1) or off (0)
  void uif_light(unsigned char on_off)
  {
    Serial1.write(0x1b);
    Serial1.write(0x42);
    Serial1.write(0x4c);
    Serial1.write(on_off);
  }
  // Set the cursor to the given x & y coordinates
  void uif_locate(unsigned char x, unsigned char y)
  {
    Serial1.write(0x1b);
    Serial1.write(0x4c);
    Serial1.write(x);
    Serial1.write(y);
  }
  // Turn the UIF-5K’s LED on (1) or off (0)
  void uif_swled(unsigned char on_off)
  {
    Serial1.write(0x1b);
    Serial1.write(0x45);
    Serial1.write(on_off);
  }
  // Turn one of the UIF-5K’s button’s LEDs on (1) or off (0)
  void uif_led(unsigned char number, unsigned char on_off)
  {
    Serial1.write(0x1b);
    Serial1.write(0x46);
    Serial1.write(number);
    Serial1.write(on_off);
  }
  // Turn the buzzer on (1) or off (0)
  void uif_buzzer(unsigned char on_off)
  {
    Serial1.write(0x1b);
    Serial1.write(0x5a);
    Serial1.write(on_off);
  }
  // Clear the display
  void uif_clear()
  {
    Serial1.write(0x1b);
    Serial1.write(0x43);
  }
}
```
Interfacing to a Character LCD (CLCD)

Connect the FA-DUINO to the CLCD via RS-232 as shown in the image below. Set all the dip switches on the CLCD to the ON position. The baudrate should be 115200.

```c
void setup()
{
  Serial1.begin(115200);               // baud rate 115200
  clcd_clear();                        // clear the screen
  delay(20);
  // Position the cursor
  clcd_locate(0,0);
  Serial1.print("== CLCD Test ==");
  delay(100);
  clcd_locate(2,1);
  Serial1.print(" FA-DUINO-24RA ");
  delay(100);
}

void loop()
{
}
```

For more information please see the ML-THRT1 user's manual.

Interfacing to the CN-RS235485

The CN-RS232485 can be used to convert the FA-DUINO's RS-232 signal to RS-485.

Interfacing to the ML-THRT1

The ML-THRT1 can be used to measure temperatures from -180~500℃ through a PT100 resistance thermometer.