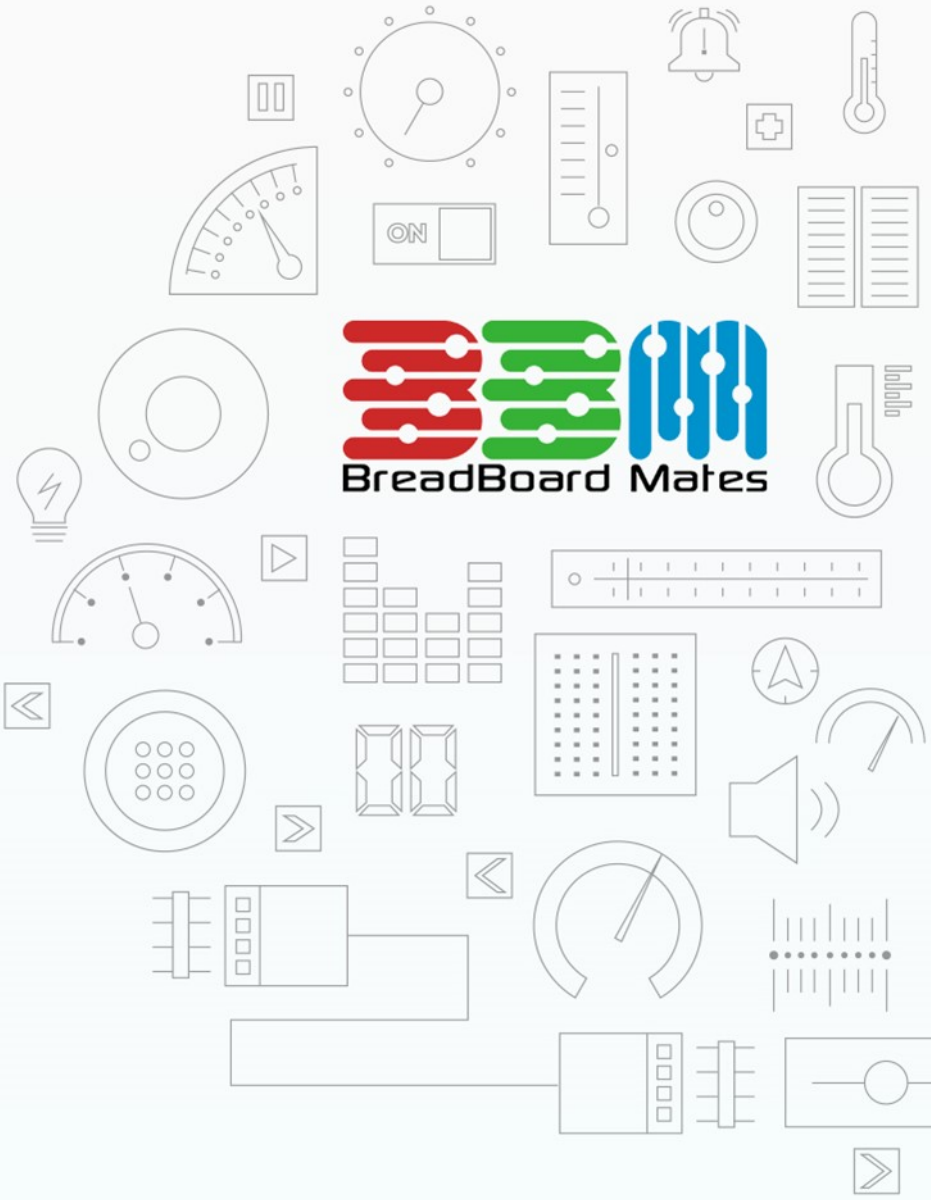


PROJECT

Send Hex Values from Arduino Serial Monitor

Content may change at any time. Please refer to the resource centre for latest documentation.

© 2024 BreadBoard Mates. All rights reserved.

Contents

Introduction

Arduino's are very common microcontroller boards used to study and design programmable electronics. It is often used with multiple peripherals such as buttons, sliders, sensors and motors.

Together with a TIMI acting as a small fancy display, Arduino boards become a lot more powerful and interesting to use in prototyping.

This project showcases a TIMI-96 module controlled by an Arduino Uno to print user inputted hex strings to a PrintArea widget.

Requirements

To proceed with the project, the following are required.

Hardware

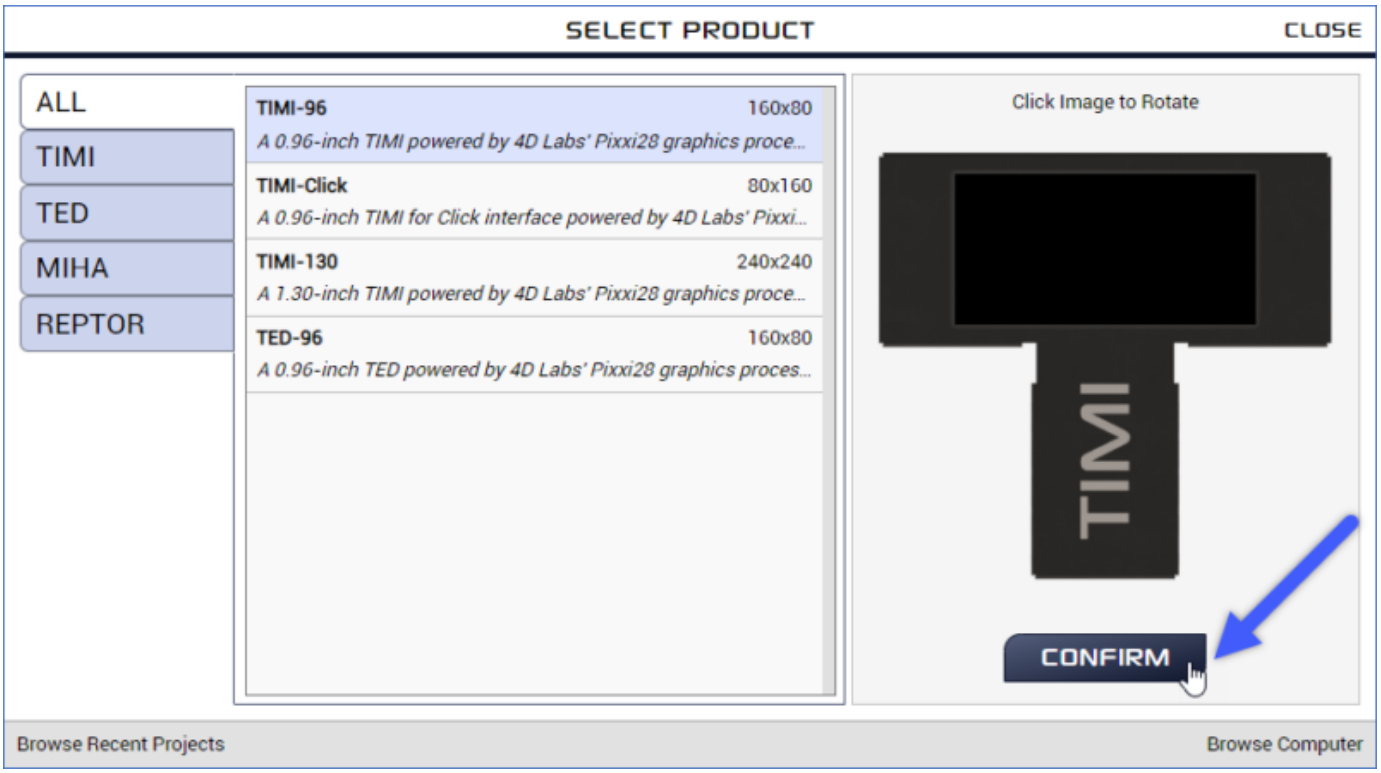
- TIMI-96
- Mates Programmer
- USB Type A to microUSB cable (for the Mates Programmer)
- USB Type A to Type B cable (for the Arduino, replace as necessary)
- Connecting Wires
- Arduino Wires
- Breadboard

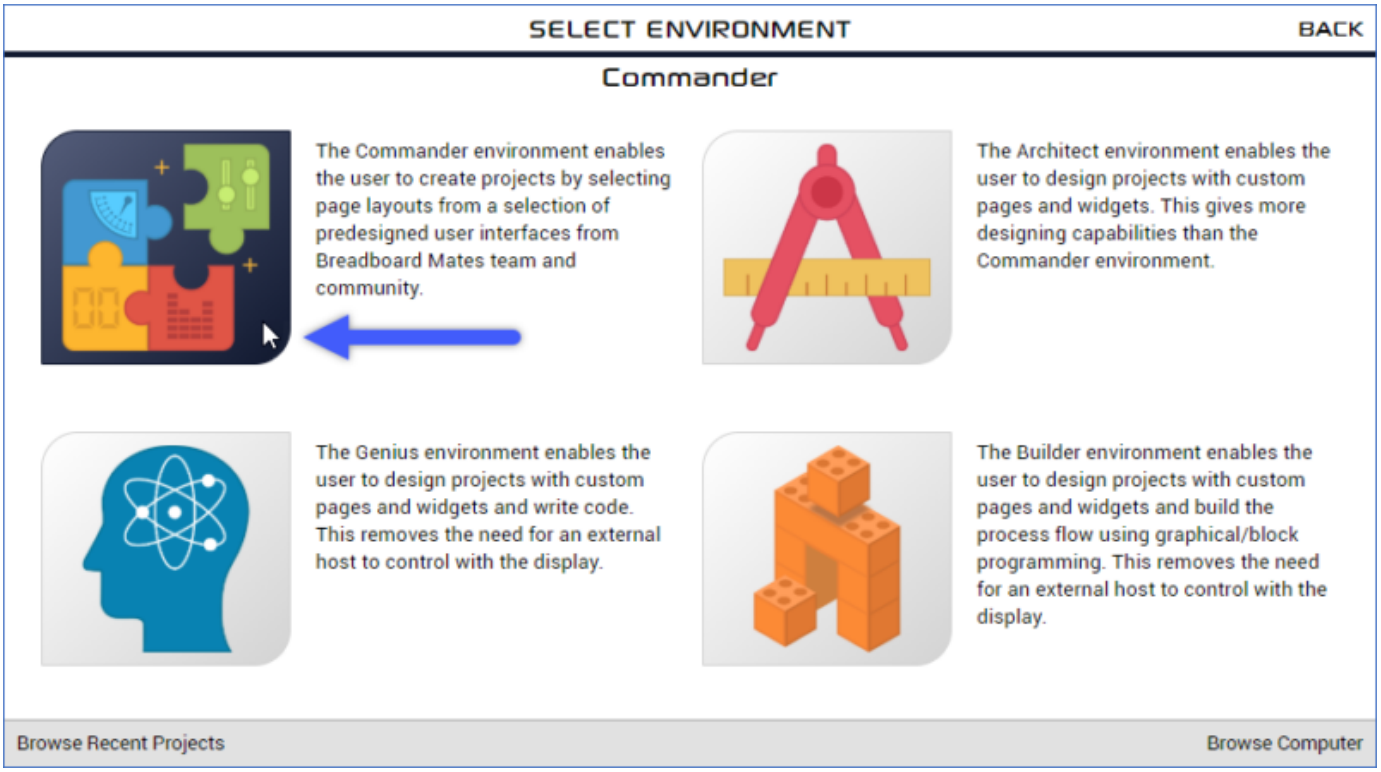
Software

- Mates Studio
- Arduino IDE

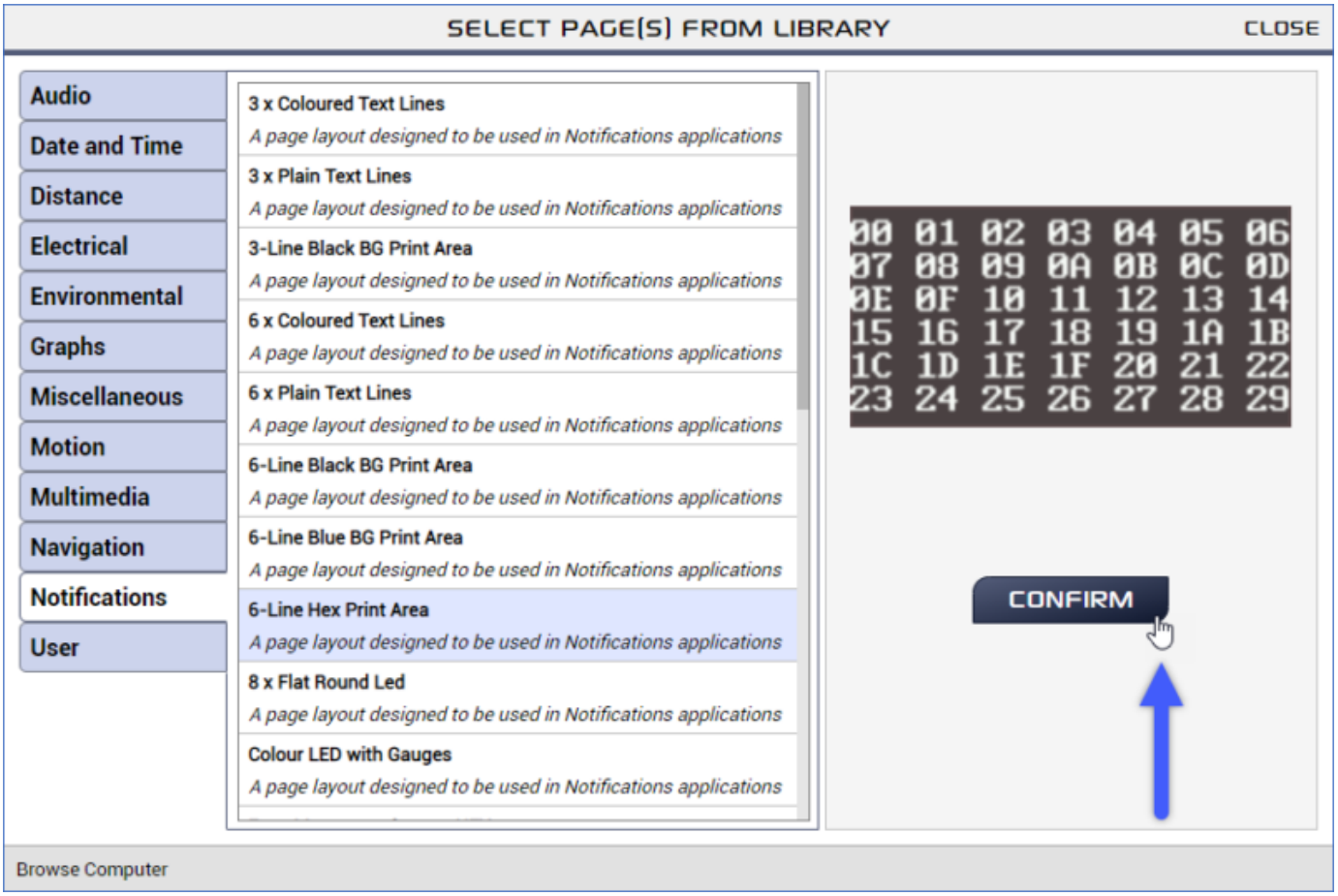
Graphics Design

Step 1: Open *Mates Studio* and create a *Commander* project for *TIMI-96* with *Reversed Landscape* orientation

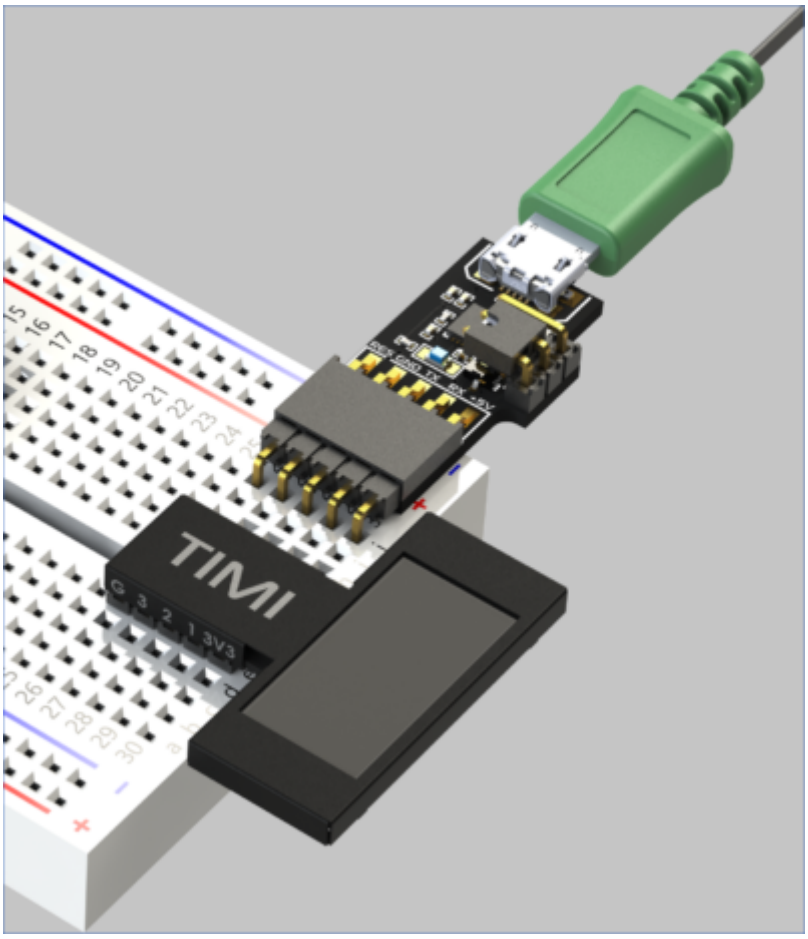




Step 2: Browse the library for appropriate page designs. For this project, *6-Line Hex Print Area* page under *Notifications* category was used.

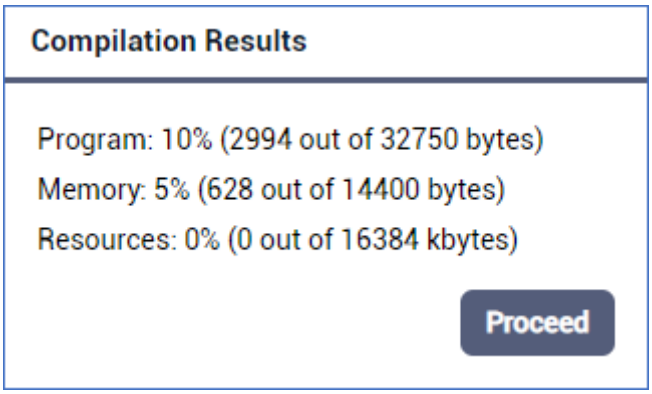


Step 3: After finalizing the design, connect TIMI-96 to your computer



Step 4: Upload the project to the appropriate COM port

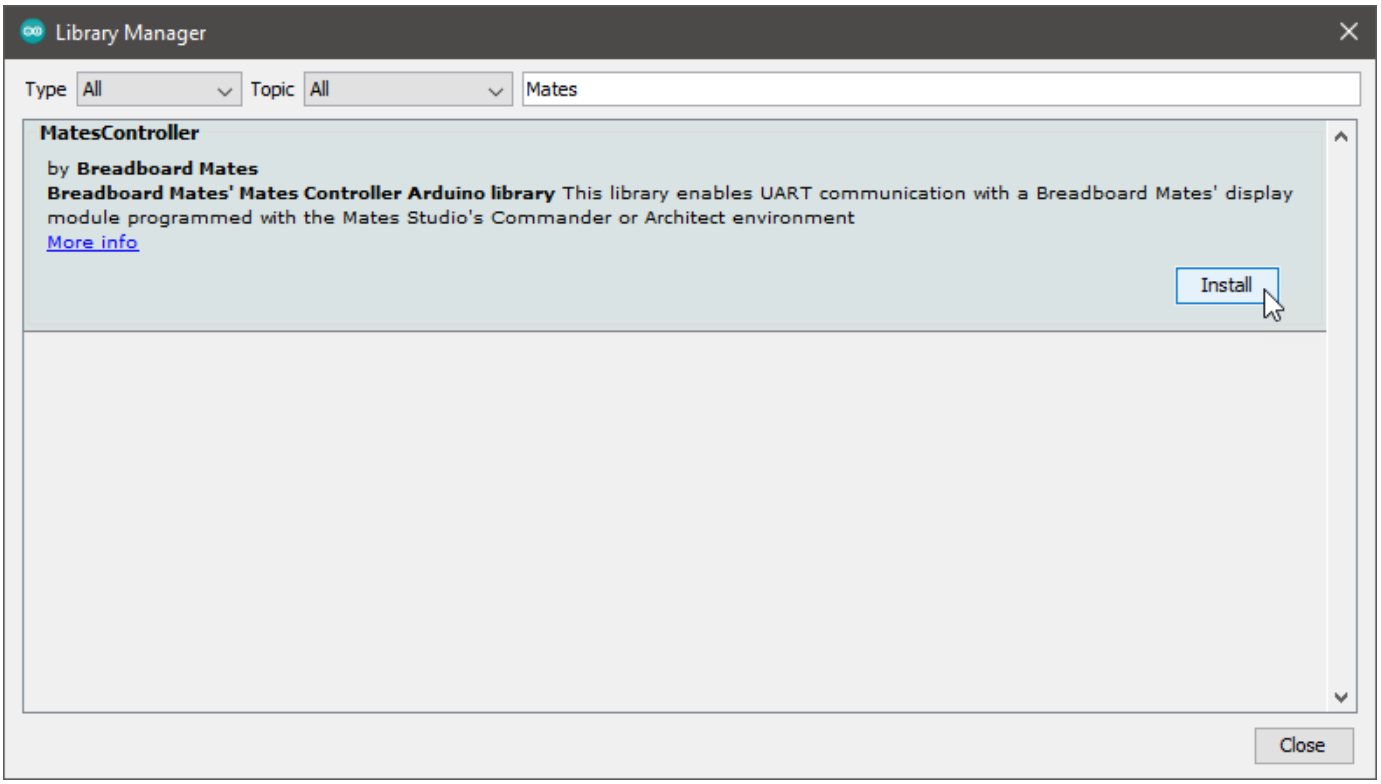
Step 5: When prompted, click *Proceed* to continue with upload.



> **Note**
>
> It is recommended that the graphics design is finalized before moving to the next steps when working on a project

Programming the Arduino

Step 1: Install the *MatesController* library using Arduino's *Library Manager*.



Step 2: Include *MatesController.h* and *SoftwareSerial.h* to your project.

```
#include "MatesController.h"
#include <SoftwareSerial.h>
```

Step 3: Create a *Software Serial* instance named *matesSerial* and declare Serial as *debugSerial*

```
#define debugSerial Serial
SoftwareSerial matesSerial(2, 3);
```

Step 4: Create a *MatesController* instance named *mates* setting *matesSerial* for TIMI and *debugSerial* for Arduino's Serial Monitor.

```
MatesController mates = MatesController(matesSerial, debugSerial);
```

This will initialize the *MatesController* instance to the default reset pin 4 using a LOW pulse.

Step 5: (Optional) Create a function for toggling the built-in LED of the Arduino board. This can be used for debugging or showing errors if the Serial monitor can't be used.

```
int errLedStatus = LOW;
void ErrorLed_Toggle() {
  errLedStatus = ~errLedStatus;
```

```
  digitalWrite(LED_BUILTIN, errLedStatus);
}
```

Step 6: (Optional) At the beginning of the setup function, set the built-in LED pin to OUTPUT and set it to LOW.

```
pinMode(LED_BUILTIN, OUTPUT);
digitalWrite(LED_BUILTIN, errLedStatus);
```

Step 7: To start using the MatesController instance, use the `begin` function

```
matesSerial.begin(9600);    // this should match Mates Studio project baudrate
debugSerial.begin(115200);  // Serial Monitor needs to be opened in this baudrate
mates.begin();
```

As shown, the Serial objects for both TIMI and Serial Monitor needs to be initialized before the mates instance.

> **Note**
>
> Debug streams needs to be setup beforehand. Serial for the BBM module also needs to be setup initially if it is not a HardwareSerial (ex. Software Serial, Alt Soft Serial)

Step 8: (Optional) The `begin` function can be enclosed in an if condition to handle initialization errors.

```
matesSerial.begin(9600);    // this should match Mates Studio project baudrate
debugSerial.begin(115200);  // Serial Monitor needs to be opened in this baudrate
if (!mates.begin()) {
  // Display didn't send ready signal in time
  while (1) {
    ErrorLed_Toggle();
    delay(100);
  }
}
```

Step 9: Create supporting functions for parse hex string to bytes

```
bool hexStringToBytes(int8_t * output, const char * hexString) {
  int len = strlen(hexString);

  if (len % 2 != 0) {
    debugSerial.println("Error: Invalid HEX string length!");
    return false;
  }

  len /= 2;
  int8_t temp;
  for (int i = 0; i < len; i++) {
    if (!hexCharToNibble(&temp, hexString[2 * i])) return false;
    output[i] = temp << 8;
    if (!hexCharToNibble(&temp, hexString[(2 * i) + 1])) return false;
    output[i] |= temp;
  }

  return true;
}

bool hexCharToNibble(int8_t * output, char hexChar) {
  if (hexChar >= '0' && hexChar <= '9') {
    *output = hexChar - '0';
  }
  else if (hexChar >= 'A' && hexChar <= 'F') {
    *output = hexChar - 'A' + 10;
  }
  else if (hexChar >= 'a' && hexChar <= 'f') {
    *output = hexChar - 'a' + 10;
  } else {
    debugSerial.println("Error: Invalid HEX character!");
```

```
    return false;
  }
  return true;
}
```

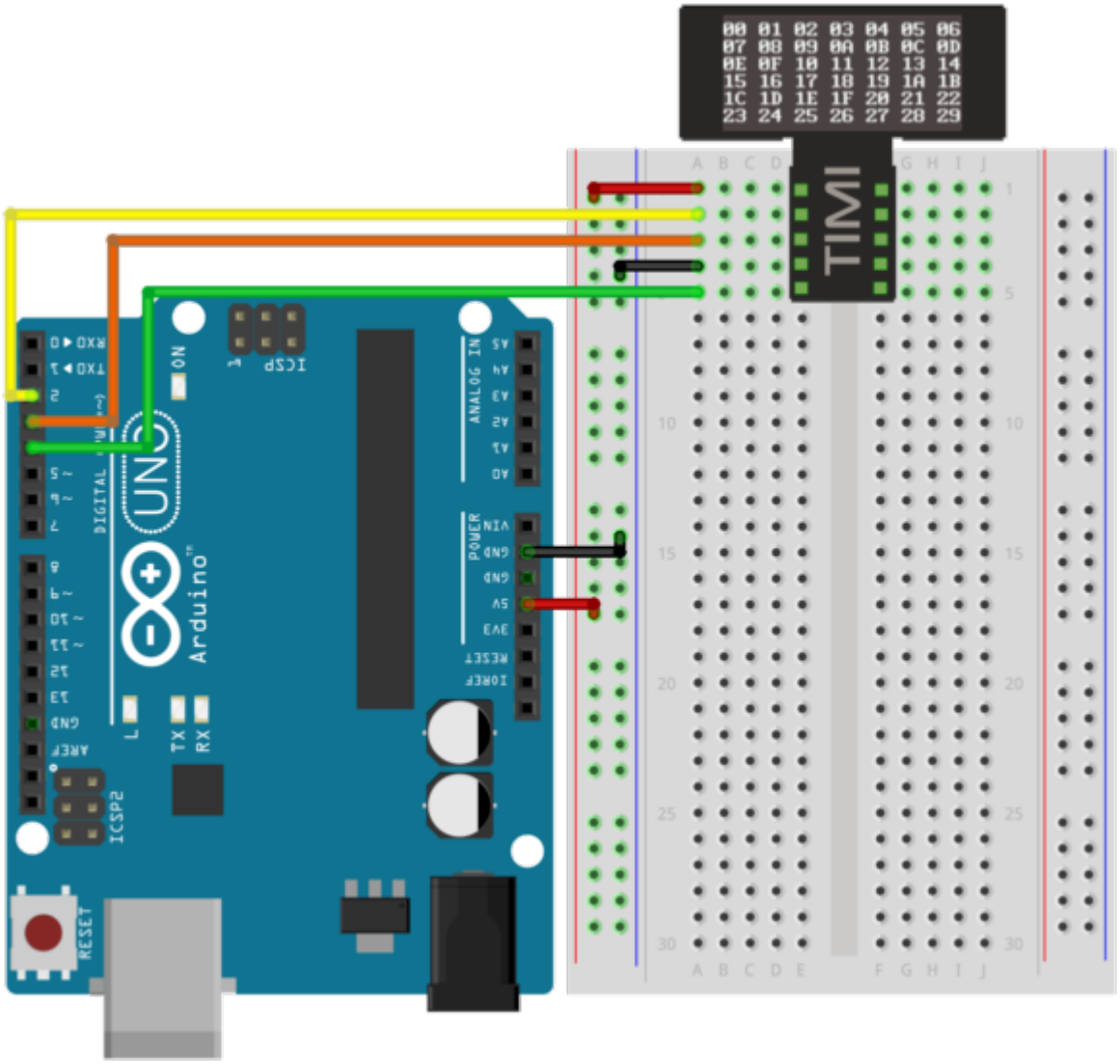
Step 10: In the loop function, the values are read from Serial monitor and sent to TIMI as necessary.

```
void loop() {
  String msg = debugSerial.readStringUntil('\n');
  int len = msg.length();

  if (len > 0) {

    len /= 2;
    int8_t buf[len];
    int8_t ctr = 0;

    debugSerial.println(msg);

    if (hexStringToBytes(buf, msg.c_str())) {

      mates.clearPrintArea(0);

      for (uint8_t i = 0; i < len; i++) {
        mates.appendToPrintArea(0, &buf[i], 1);
        delay(100);
        ctr++;
        if (ctr >= 42) {
          delay(900);
          mates.clearPrintArea(0);
          ctr = 0;
        }
      }

      debugSerial.println("Done!");

    }

    debugSerial.println();
    debugSerial.print("Enter HEX (w/o spaces): ");
  }
}
```

As shown, the code starts by reading the Serial monitor for user input. If there is any, the code attempts to parse the hex string to an 8-bit array. If successful, the PrintArea is cleared and the hex bytes are printed sequentially.

Running the Project

After designing the user interface for TIMI and writing code for the Arduino and programming them, it is time to connect the devices together. Follow the diagram below for the connection between TIMI and Arduino.



Finally, supply power to the Arduino and observe the behavior of the project.

Downloadable Resources

- Mates Studio
- Arduino IDE
- Arduino Mates Controller Library
- Project Files