M5Stack Tab5 + INA3221 Current Sensor

by Bryan A. “CrazyUncleBurton” Thompson

Last Updated 2/21/2026

Sensor

Adafruit INA3221 Triple Current / Voltage Sensor

Connections

1. Connect the USB C Cable – USB C on the Tab5 end and whatever type will plug into your computer on the other end.
2. Connect the 4 pin plug end of the sensor cable to red port on M5Stack Tab5.
3. Connect the 4 wire Grove to Dupont connector cable to the pins on the INA3221 sensor:

Sensor	Cable/Port
VCC	Red/VCC
GND	GND
SDA	Yellow/SDA
SCL	White/SCL

The Platform IO Project

The Platform IO project is located here: https://github.com/CrazyUncleBurton/M5Stack-Tab5-and-TI-INA3221-Current-Sensor

See the document “VS Code with Source Control.pdf” located in the same directory as this document.

Testing

1. Connect a USB cable from the computer that VS Code is running on to the M5Stack microcontroller.
2. If the microcontroller is not on, press the white button on the side once to turn it on.
3. In VS Code, click the Platform IO Upload icon  in the bottom toolbar. The hover text will be Platform IO: Upload.
4. The project will compile. When it’s ready to upload, we see lines like “Connecting…” and “Writing…”. Wait for the line that says SUCCESS in green text before proceeding.
5. When finished, the microcontroller will reboot.
6. When the microcontroller reboots, watch the display. You should see a one-time splash screen that says **M5Stack Tab5 Starting…** and then **INA3221 Found**
7. After about one second (a delay inserted to let the user see the splash screen), we see the first update from the current and voltage sensors.
8. One second later (again, to read the screen), it refreshes the data and updates the screen.

Program Explanation

We started with the program from the manufacturer as an example for using their sensor. It only output data to the LCD.

We added two libraries – one provided by the Microcontroller manufacturer and one provided by the Sensor manufacturer:

```
#include <M5Unified.h>
#include "Adafruit_INA3221.h"
```

We added some variables to store text size and formatting variables:

```
int lineHeight = 50;
int x_margin = 25;
int y_header = 25;
int x = x_margin;
int y = y_header;
```

We added some constants to store I2C pins:

```
const int i2cSdaPin = 53; - Wire() is connected to the external bus on these pins
const int i2cSclPin = 54; - Wire() is connected to the external bus on these pins
```

We create an instance / object of the Adafruit_INA3221 library called ina3221:

```
Adafruit_INA3221 ina3221;
```

We create an instance / object of the M5GFX video driver library (part of M5Unified library) with the name "display":

```
M5GFX& display = M5.Display;
```

We create an instance / object of the M5Canvas library (part of the M5Unified library) this line to create a canvas named "canvas":

```
M5Canvas canvas(&display);
```

The **setup()** function runs once at microcontroller power-up and restart.

These lines start the M5Unified library which supports the microcontroller:
auto cfg = M5.config();
M5.begin(cfg);

This line starts the Arduino Wire(), which is the external I2C bus where the sensor is connected:
Wire.begin(i2cSda, i2cScl);

These lines configure the display and canvas:
display.setRotation(1); - This sets which way is up on the LCD.
canvas.setColorDepth(16); - This sets the number of bits of color info on the LCD.
canvas.createSprite(display.width(), display.height()); - These lines create the canvas, which is the background area where we build the screen. It won't be displayed until later.
canvas.setTextColor(TFT_WHITE, TFT_BLACK); - This sets text color to white, and black background color to white.
canvas.setFont(&fonts::efontJA_16_b); - This is a built-in font that supports Unicode symbols
canvas.setTextSize(textSize); - This sets the size of text we are printing to the screen.
canvas.setTextWrap(false, false); - This says not to wrap the text if it prints off the end of the line

These lines draw the initial splash screen:
canvas.fillScreen(TFT_BLACK); - clear the canvas
y=200; - text position for the splash screen chosen to be approx. centered on the screen
x=475;
canvas.setCursor(x, y); - start writing to the canvas at the new coordinates
canvas.printf(" M5Stack Tab5");
y=275; - Update y to the next row
canvas.setCursor(x, y); - start writing to the canvas at the new coordinates
canvas.printf(" Starting...");
if (!ina3221.begin(0x40, &Wire)) { - Initialize the library and test if the sensor is connected - the sensor's address on the I2C bus is 0x40 and it is on the external I2c BUS
  y=425;
  canvas.setCursor(x, y);
  canvas.printf("Failed to find INA3221 chip"); - if we get to here, we didn't find the sensor
  canvas.pushSprite(0, 0);
  while (1)
   delay(10); - wait 10mS and try the sensor again
}
y=425;
canvas.setCursor(x, y);
canvas.printf("INA3221 Found!"); - if we get to here, we found the sensor
y += lineHeight;

These lines configure the operating mode of the sensor:

```
ina3221.setAveragingMode(INA3221_AVG_16_SAMPLES); - smooth data from the sensor
for (uint8_t i = 0; i < 3; i++) {
   ina3221.setShuntResistance(i, 0.05); - tell the sensor about our shunt
}
```

These lines publish the canvas to the LCD:

```
canvas.pushSprite(0, 0); - send the canvas to the LCD
delay(3000); - wait to let the user read the screen
```

This is the end of the one-time setup() function.

The function drawDashboard() creates the dynamic canvas with sensor data:

y = y_header; - space above the first row.
x = x_margin; - margin to the left of text
canvas.setCursor(x, y); - move cursor to top left position
canvas.fillScreen(TFT_BLACK); - clear screen
canvas.setTextSize(textSize); - set size of text to write

for (uint8_t i = 0; i < 3; i++) { - loop to read all the sensor data as long as i < 3. When i=3, stop looping and proceed to the next instruction after the loop.

float voltage = ina3221.getBusVoltage(i); - reads sensor voltage on channel i and assign to floating variable voltage

float current = ina3221.getCurrentAmps(i) * 1000; - reads sensor current on channel i and convert to mA and assign to floating variable current.

canvas.setCursor(x, y); - move the cursor to the next line

canvas.printf("Channel %d Voltage = %.2fV Current = %.2fmA", i, voltage, current); - This line writes a line to the canvas containing the following:

- "Channel" and then the contents of the "i" variable which is the loop index and contains the current channel we're reading, formatted as an integer.
- " Voltage =" and then the voltage measurement, formatted as a floating point with two digits after the decimal, and then "V" (the units of our voltage measurement)
- "Current = " and then the current measurement, formatted as a floating point with two digits after the decimal, and then "mA" (the units of our current measurement)

y += lineHeight; // before we loop, update y to write to the next line on the LCD

}

That's the end of our loop. Now let's update the screen:

canvas.pushSprite(0, 0); // Once we gather the data and create the canvas, make the data live.

y=y_header; // Reset y to the start of the screen for next update

Now we're running in the main() loop:

This code causes the main loop to run once every second:

```
static uint32_t lastFrameMs = 0;
const uint32_t frameIntervalMs = 1000;
if (millis() - lastFrameMs < frameIntervalMs) {
  return;
}
lastFrameMs = millis();
```

This calls the function which creates the canvas and then publishes the canvas:

```
drawDashboard();
```

When we get to the end of the list, we loop back to the start of the main loop and run it again. The program doesn't end.

Reference

Tab5 Microcontroller Info: https://docs.m5stack.com/en/core/Tab5

Sensor Info: https://learn.adafruit.com/adafruit-ina3221-breakout

M5GFX Display Library Stuff: https://docs.m5stack.com/en/arduino/m5gfx/m5gfx_functions

M5Canvas Stuff: https://docs.m5stack.com/en/arduino/m5gfx/m5gfx_canvas