```
/ /August 23, 2025, Triggered Oscilloscope
#include <MCUFRIEND_kbv.h>
#include <Adafruit_GFX.h>

// Color definitions
#define BLACK   0x0000
#define WHITE   0xFFFF
#define GREEN   0x07E0
#define RED     0xF800
#define GRAY    0x8410

MCUFRIEND_kbv tft;

const int graphTop = 5;
const int graphHeight = 300;
const int graphLeft = 5;
const int graphWidth = 450;

// Define various ADC prescaler, assigns a 1 to ADPSx bit, example (1<ADPS_16) sets bit 2 to a
1 ie register looks like xxxxx100 doesn't make a lot of sense

const unsigned char PS_16 = (1 << ADPS2);
const unsigned char PS_32 = (1 << ADPS2) | (1 << ADPS0);
const unsigned char PS_64 = (1 << ADPS2) | (1 << ADPS1);
const unsigned char PS_128 = (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0);

int prevY = 0;
int xPos = graphLeft;
int analogValue;
int oldAnalogValue;

bool block;
void drawAxes();
void clearGraphArea() ;
```

```
void setup()
{
  Serial.begin(9600);
  uint16_t ID = tft.readID();
  tft.begin(ID);
  tft.setRotation(1);
  tft.fillScreen(BLACK);

  // set up the ADC
  ADCSRA &= ~PS_128; // remove bits set by Arduino library
  // you can choose a prescaler from below.
  // PS_16, PS_32, PS_64 or PS_128
  ADCSRA |= PS_64; // set our own prescaler and speeds up conversion time at the expense of
accuracy
  drawAxes();
}
```

```
//**************start of Looping*********************************
void loop()
{
  int y;
  // Read analog input
  analogValue = analogRead(A6); // 0 - 1023
  if (block==false) //ensure entering triggering point once at exactly right point on wave
  {
    //if (analogValue ==400 && analogValue > oldAnalogValue) // uncomment for triggering
    if (analogValue >0) // free running, comment for triggering
    {
            y = map(analogValue, 0, 1023, graphTop + graphHeight, graphTop);
           // changes the analog input range from 0 to 1023, to 305, 5
            // Draw line from previous point
           if (xPos > graphLeft)
             {
                       tft.drawLine(xPos - 1, prevY, xPos, y, GREEN);
             }
      prevY = y;
      xPos++;
      block=true;
    }
  }
   else if(block==true) // after triggering continue plotting until end of screen
  {
      y = map(analogValue, 0, 1023, graphTop + graphHeight, graphTop);
      // changes the analog input range from 0 to 1023, to 305, 5
      // Draw line from previous point
      if (xPos > graphLeft)
         {
           tft.drawLine(xPos - 1, prevY, xPos, y, GREEN);
         }

    prevY = y;
    xPos++;
  }

  // Scroll graph when end is reached
  if ((xPos >= graphLeft + graphWidth))
  {
    block=false;
    xPos = graphLeft;
    prevY = map(analogRead(A6), 0, 1023, graphTop + graphHeight, graphTop);
    clearGraphArea();
    drawAxes();
  }
 oldAnalogValue=analogValue;
```

```
 delayMicroseconds(1000); // Adjust sample rate
}
//********************end of looping*********************************
```

```
// Draw X and Y axes with labels and grid lines
void drawAxes()
{
  tft.setTextSize(1);
  tft.setTextColor(WHITE);

  // Y axis
  tft.drawLine(graphLeft, graphTop, graphLeft, graphTop + graphHeight, WHITE);
  //(20,40,20,40+160)

  for (int i = 0; i <= 5; i++)
          {
           int y = graphTop + i * (graphHeight / 5);
           tft.drawLine(graphLeft, y, graphLeft + graphWidth, y, GRAY);  // horizontal grid
            tft.setCursor(0, y - 5);
           tft.print(5- i );
          }

  // X axis line
  tft.drawLine(graphLeft, graphTop + graphHeight, graphLeft + graphWidth, graphTop +
graphHeight, WHITE);
  tft.setCursor(graphLeft + graphWidth + 5, graphTop + graphHeight - 8);
  tft.print("time");
  for (int i = 0; i <= graphWidth; i=i+50)
           {
            tft.drawLine(graphLeft+i, graphTop, graphLeft+i, graphTop + graphHeight, WHITE);
          //(20,40,20,40+160)
          tft.setCursor(i, graphTop + (graphHeight));
           tft.print(i);
           }
}

// Clears the graph area only (not the whole screen)
void clearGraphArea()
{
  delay(1000);
  tft.fillRect(graphLeft + 1, graphTop + 1, graphWidth - 1, graphHeight - 1, BLACK);

}
```