Analog X-Y Pen Plotter Output To HP-GL Translator
http://github.com/jshorstman/Analog-XY-Plotter-Output-to-HPGL-Translator

What it is:
An analog X-Y pen plotter output to HP-GL translator. (Hewlett-Packard Graphics Language (http://en.wikipedia.org/wiki/HPGL)

What it does:
Some measurement devices, like the Tektronix 2232 Digital Storage Oscilloscope, have an analog X-Y pen plotter output port so that the display image can be captured to an analog X-Y pen plotter or recorder. See oscilloscope display, below.

The problem is one wishes to capture the display image in digital form onto a modern day personal computer, since analog pen plotters are virtually extinct. PC images can be pasted directly into documents.

The 'Analog X-Y Pen Plotter Output To HP-GL Translator' will continually sample the X and Y analog voltages from the plotter port, digitize them, and package the values into a series of plain text HP-GL plotter commands, which are output via a USB port to a file on a PC. The HP-GL file can be processed into a display image by any commonly available plotter emulator application, such as PrintCapture (http://www.printcapture.com). See actual plot image, below.

What it's made of:
1. Arduino UNO for the ADC conversions, digital filtering, assembly of HP-GL commands and serial output.
2. Arduino 'Shield' type board for signal conditioning (voltage translation), voltage reference and power monitoring. See schematic and boards, below.
3. DB9 cable for connection to the oscilloscope plotter port. See Auxiliary Connector, below.
4. USB cable between Arduino UNO and PC.
5. 9VDC AC adapter (a wall wart).

What's on the Arduino shield board:
The Arduino shield board contains circuitry comprised of (see schematic);

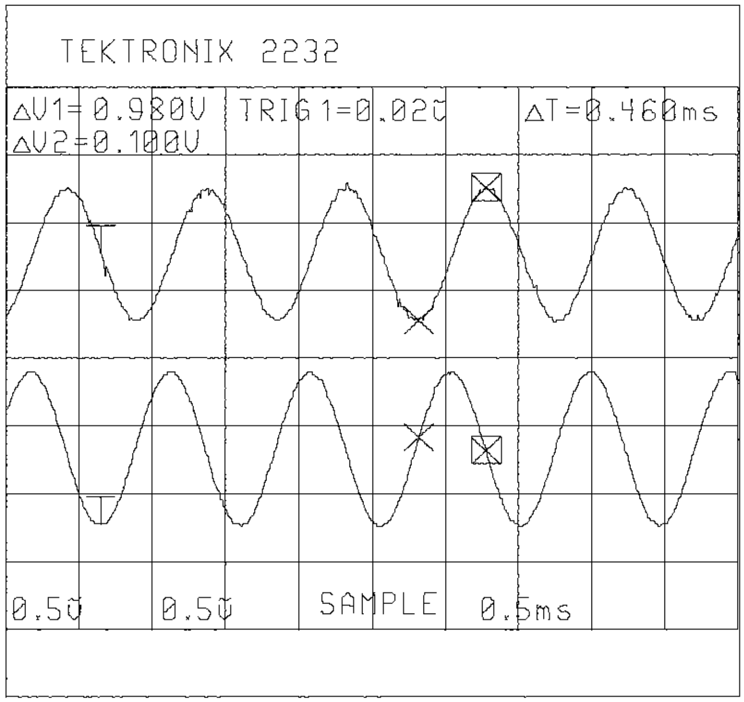
1. Two (2) precision, unity-gain differential amplifiers (wired as summing amps) to translate the **±2.5V** analog X and Y voltages into **0 to 5V** for the Arduino analog pins. U2, U3
2. A 'charge pump' voltage converter to convert 9VDC to ±9VDC to power the amplifiers. U1, C1, C2
3. A precision 2.50V voltage reference to the amplifiers for the voltage translation. U4
4. A precision 5.00V voltage reference to the Arduino AREF pin. U5
5. A voltage divider network so the 9V Vin can be reduced to 5V for the Arduino analog pin. A schottky diode protects the pin if a higher voltage power supply is used. R1, R2, D1
6. A bi-color red/green LED as a Go/No Go indicator that the 9V power supply is connected, or not. The Arduino sketch will stay in the setup() routine until a 9VDC, or greater, power supply is plugged into the Arduino UNO. These IC's cannot run accurately on the USB ~5V power! LED, R3
7. A slide switch to start the data stream and stop it when the plot is completed. S1
8. A pair of capacitors to filter the DAC dithering of the plotter port X and Y voltage signals. C9, C10
9. A 10-pin header for connection to the oscilloscope plotter port. J1

Operation:

1. Slide switch to OFF.
2. Connect Arduino via DB-9 connector to oscilloscope plotter port.
3. Connect Arduino USB to PC. Arduino powers up, LED is red.
4. Connect 9VDC power to Arduino. LED is green.
5. Launch listening application, like TeraTerm or PrintCapture. Ensure your application is set to the Arduino UNO COM port 6, 115200, 8, none, 1.
6. Slide switch to ON. (data stream commences)
7. Immediately press PLOT key on scope.
8. When scope plot completes immediately slide switch to OFF. (data stream halts)
9. Reset Arduino in order to start the next plot.

Example HP-GL output to file:

```
IN;
PS8900,8900;
IP0,0,8900,8900;
SC0,250,0,1023,1;
PU;          // Pen Up
SP1;
PA77,984;    // Plot Absolute to (77,984)
...          // more data points (PA) every 1.6 milliseconds
PD;          // Pen Down
PA53,92;
...          // more data points (PA) every 1.6 milliseconds
PU;
SP;
IN;
```

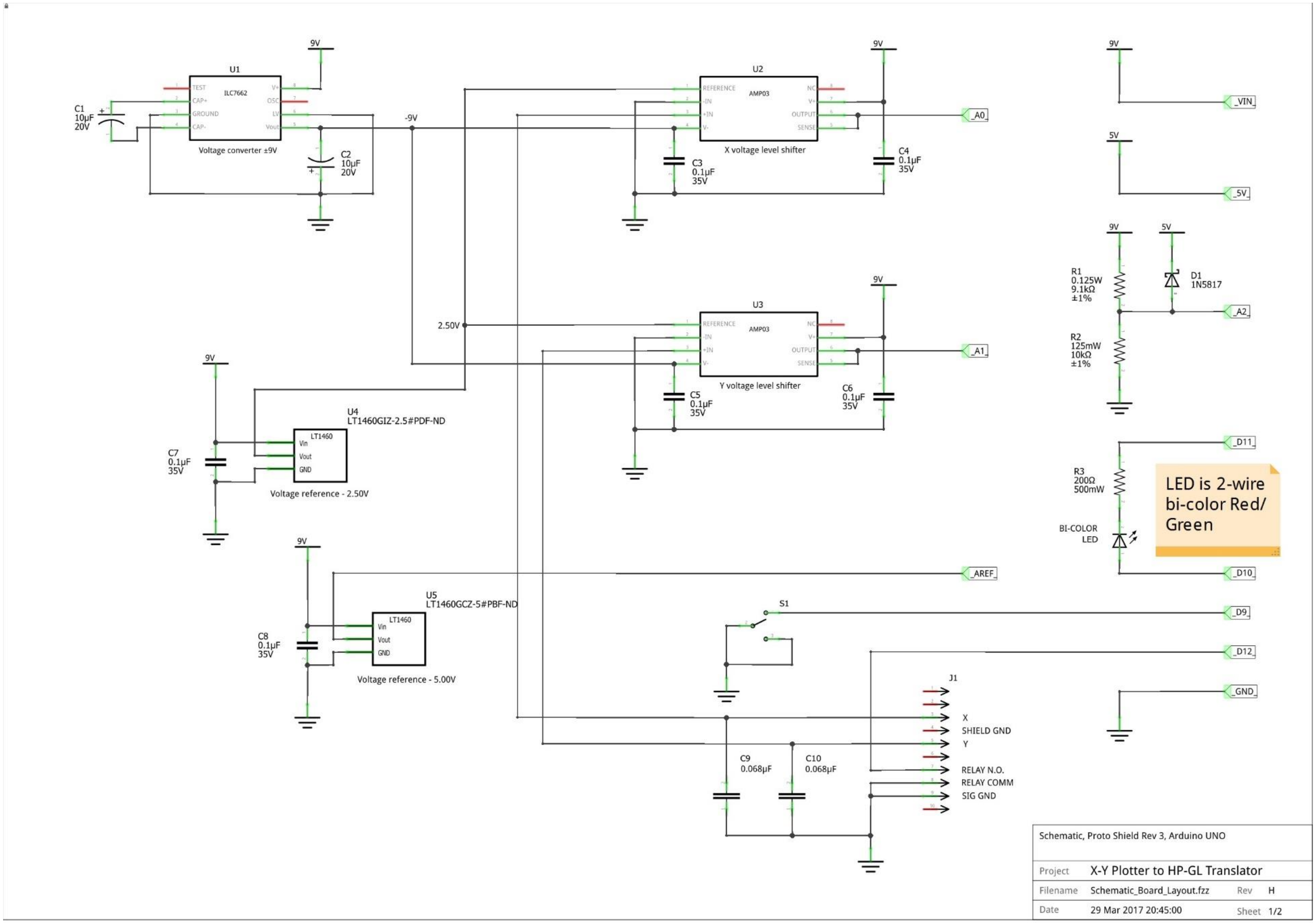


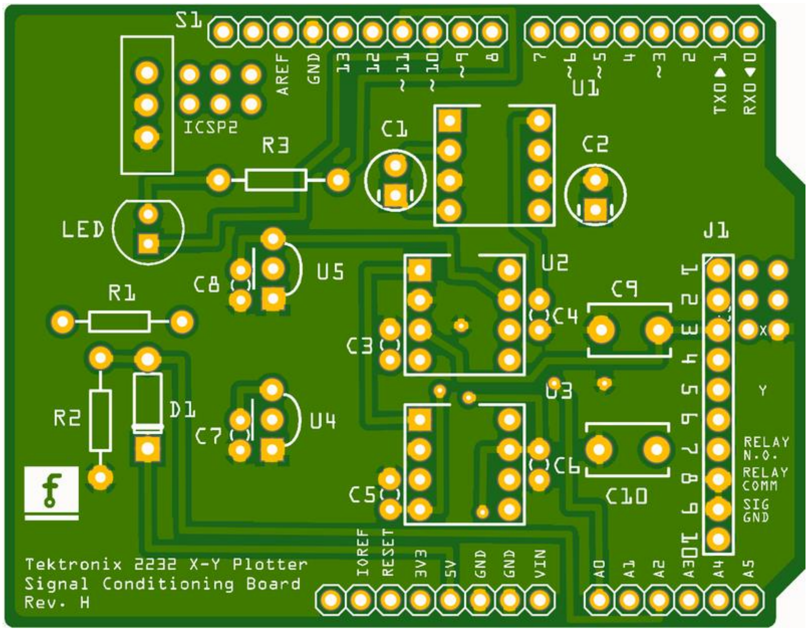
Actual plot image from an HP-GL file created by the Translator of a Tektronix 2232 oscilloscope display. Image created by PrintCapture.

Actual Tektronix 2232 oscilloscope display.

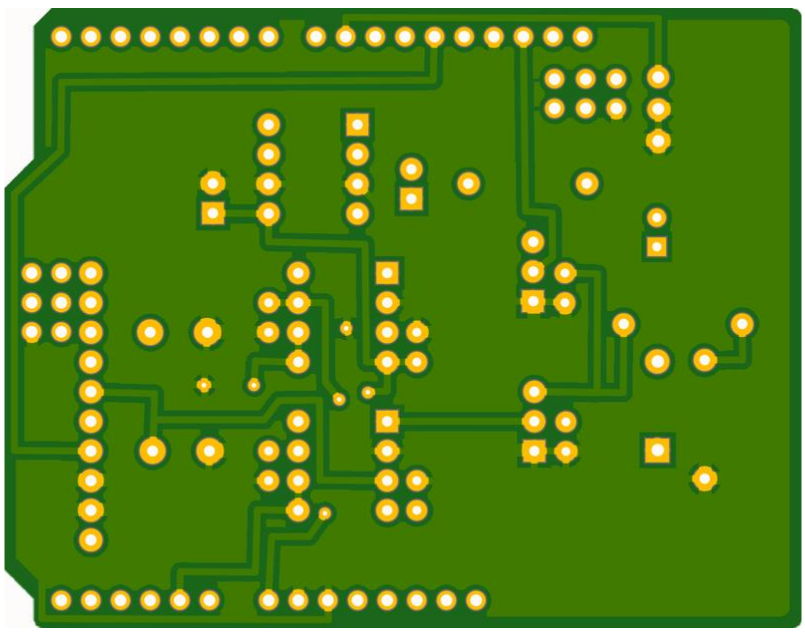
Analog X-Y Pen Plotter Output To HP-GL Translator
http://github.com/jshorstman/Analog-XY-Plotter-Output-to-HPGL-Translator

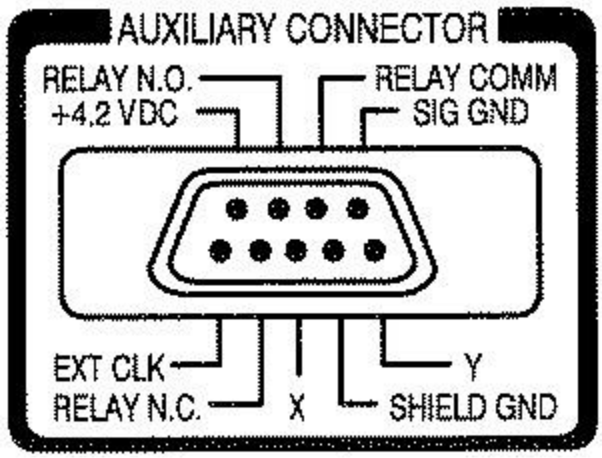
Schematic:





Arduino shield board front


Arduino shield board back



Tektronix Auxiliary Connector for X-Y plotter. Relay is for pen up/down

Arduino Sketch:

```
/*
 ***************************************************************************************|
 * Tektronix 2232 Digital Storage Oscilloscope analog X-Y pen plotter to HP-GL
 * translator. (Hewlett-Packard Graphics Language http://en.wikipedia.org/wiki/HPGL)
 *
 * John Horstman
 * 3/19/2017 Rev H
 *
 * This sketch continuously reads the X and Y analog pen plotter voltages, and the
 * pen up/down signal, and packages the ADC values in successive HP-GL commands. The
 * commands can be serially output to a file, via TeraTerm, or to a plotter application
 * like PrintCapture at http://www.printcapture.com/.
 *
 * The HP-GL commands used in this sketch are:
 * IN;  Initialize
 * PS:  Plot size
 * IP;  Input P1 & P2
 * SC;  Scale
 * SP;  Select pen
 * PU;  Pen UP
 * PD;  Pen DOWN
 * PA;  Plot Absolute
 *
 * Example HPGL output of this sketch:
 * IN;
 * PS8900,8900;
 * IP0,0,8900,8900;
 * SC0,250,0,1023,1;
 * PU;
 * SP1;
 * PA77,984;
 * ...          // data points every 1.6 milliseconds
 * PD;
 * PA53,92;
 * ...
 * PU;
 * SP;
 * IN;
 *
 * This sketch is for Arduino UNO with Proto Shield. The Proto Shield contains signal
 * conditioning circuitry comprised of;
 * 1. Two (2) precision, unity-gain differential amplifiers (wired as summing amps)
 *    to translate the Â±2.5V analog X and Y voltages into 0 to 5V for the Arduino analog
 *    pins.
 * 2. A 'charge pump' voltage converter to convert 9VDC to Â±9VDC to power the amplifiers.
 * 3. A precision 2.50V voltage reference to the amplifiers for the voltage translation.
 * 4. A precision 5.00V voltage reference to the Arduino AREF pin.
```

```
 * 5. A voltage divider network so the 9V Vin can be reduced to 5V for the Arduino
 *    analog pin. A schottky diode protects the pin if a higher voltage power supply
 *    is used.
 * 6. A bi-color red/green LED as a Go/No Go indicator that the 9V power supply
 *    is connected, or not. This sketch will stay in the setup() routine until a
 *    9VDC, or greater, power supply is plugged in to the Arduino. These IC's
 *    cannot run accurately on the USB ~5V power!
 * 7. A slide switch to start the data stream and stop it when the plot is completed.
 * 8. A pair of capacitors to filter the DAC dithering of the plotter X-Y voltage signals.
 * 9. A 9-pin cable with DB-9 male connector for connection to the oscilloscope plotter
 *    port.
 *
 * Operation:
 * Slide switch to OFF.
 * Connect Arduino via DB-9 connector to oscilloscope plotter port.
 * Connect Arduino USB to PC. Arduino powers up, LED is red.
 * Connect 9VDC power to Arduino. LED is green.
 * Launch listening application, like TeraTerm or PrintCapture. Ensure your application
 *  is set to the Arduino COM port 6, 115200, 8, none, 1.
 * Slide switch to ON. (data stream commences)
 * Immediately press PLOT key on scope.
 * When scope plot completes immediately slide switch to OFF. (data stream halts)
 * Reset Arduino in order to start the next plot.
 ***************************************************************************************|
*/
//#define __serial_plotter // uncomment to calibrate with Arduino Serial Plotter tool
//#define __dummy_data // uncomment to use dummy data for testing

// The pins
#define TestPin 8      // Digital pin 8
#define SwitchPin 9    // Digital pin 9
#define LEDGreenPin 10 // Digital pin 10
#define LEDRedPin 11   // Digital pin 11
#define PenInputPin 12 // Digital pin 12
#define IdleLED 13     // Digital pin 13
#define X_voltage 0    // Analog pin A0
#define Y_voltage 1    // Analog pin A1
#define VinVoltage 2   // Analog pin A2
// The numbers
unsigned long LoopTime; // Marker for loop() delay timer
unsigned int PenState = HIGH;        // Default pen up
unsigned int OldPenState = PenState; //
const unsigned int numReadings = 8;       // Must be a power of two so we can right shift later
unsigned int Xreadings[numReadings];      // Arrays for computing moving average for smoothing
unsigned int Yreadings[numReadings];      //
unsigned int Xtotal = numReadings * 511;  // Arrays will be preloaded
unsigned int Ytotal = Xtotal;             //
unsigned int readIndex = 0;               //
const unsigned int dcXOffset = 0; //21;  // Compensate for any DC offset. Adjust this to get approximately 511,511.
```

```
const unsigned int dcYOffset = 0; //23;  //
// The strings
String ADCXStr; // ADC X counts string
String ADCYStr; // ADC Y counts string
String CmdStr;  // HPGL command string

#ifdef __dummy_data
const float slew = 614.0; // max slew rate is 3.0 volts/second = 614 counts/second (@5 volts = 1023 counts)
const float Freq = (1.0/PI) * (slew / 1023.0);  // This frequency sinusoidal will have that slew rate (at the zero crossing)
const float TWO_PI_F = 2.0 * PI * Freq;
#endif

/*
 * The Setup ****************************************************************************|
 */
void setup() {
  // Set I/O pins
  pinMode(TestPin, OUTPUT); // timing test pin
  pinMode(IdleLED, OUTPUT);
  pinMode(LEDRedPin, OUTPUT);
  pinMode(LEDGreenPin, OUTPUT);
  pinMode(SwitchPin, INPUT_PULLUP);
  pinMode(PenInputPin, INPUT_PULLUP); // Pen signal is from Normally Open (N.O.) relay contacts

  analogReference(DEFAULT);

  // Set LEDs
  digitalWrite(IdleLED, HIGH); // Turn on 'idle' LED
  digitalWrite(LEDRedPin, LOW); // Turn red/green LED off
  digitalWrite(LEDGreenPin, LOW);

  // initialize serial communication:
  Serial.begin(115200);

  // Check for 9V Vin external power, stay in setup() until 9V supply connected
  /*
   * Why 900 counts? To gaurantee that a 7.5 VDC±5% power pack will not trigger the
   * 'Go' signal under maximal conditions:
   * 7.5v+5%; R1Ω-1%; R2Ω+1%; USB 5v-5% (AREF); +3 counts (2 LSB ADC error)
   */
  while (analogRead(VinVoltage) < 900) {
    digitalWrite(LEDRedPin, HIGH); // Turn LED red (no-go)
    digitalWrite(LEDGreenPin, LOW);
    delay(100);
  } // Wait here until user plugs in a 9VDC, or greater, power supply

  // Vin OK, continue
  digitalWrite(LEDRedPin, LOW); // Turn LED green (go)
  digitalWrite(LEDGreenPin, HIGH);
```

```
  //Wait here for switch to be switched on (ground the pin)
  while (digitalRead(SwitchPin) == HIGH) {
    delay(100);
  }
  // User is ready for data stream, let 'er rip!

  analogReference(EXTERNAL); // External precision 5.00V reference

  // Preset the averaging arrays
  for (int thisReading = 0; thisReading < numReadings; thisReading++) {
  Xreadings[thisReading] = 511;
  Yreadings[thisReading] = 511;
  }

  digitalWrite(IdleLED, LOW); // turn off 'idle' LED, transmission will commence (TX LED will light)

  // HP-GL plot header commands
  Serial.println("IN;"); // Initialize
  Serial.println("PS8900,8900;"); // Plot size; A-size
  Serial.println("IP0,0,8900,8900;"); // Input Scaling Points P1 & P2; A-size drawing (8.5" x 11", less margins)
  Serial.println("SC0,1023,0,1023,1;"); // Scale; Isotropic
  //Serial.println("OH;"); // Output hard clip limits
  //Serial.println("OP;"); // Output P1 & P2
  Serial.println("PU;"); // Pun Up
  Serial.println("SP1;"); // Select Pen 1
} // end setup

/*
 * The Loop *****************************************************************************|
*/
void loop() {
  LoopTime = micros(); // mark loop start

  // Read X, Y and Pen signals as close together as possible and compute a moving average for smoothing
  Xtotal -= Xreadings[readIndex]; // Subtract an old value from the total
  Ytotal -= Yreadings[readIndex]; //

  PenState = digitalRead(PenInputPin); // Read pen input; returns HIGH or LOW
  Xreadings[readIndex] = analogRead(X_voltage) - dcXOffset; // compensate for dc offset
  delayMicroseconds(16); // Delay two ADC clock cycles (there is less 'jitter' on the Y readings than for a one ADC clock cycle
dalay, don't ask me why)
  Yreadings[readIndex] = analogRead(Y_voltage) - dcYOffset;

  Xtotal += Xreadings[readIndex];   // Add the new value to the total
  Ytotal += Yreadings[readIndex++]; // Increment readIndex here
  readIndex %= numReadings;  // Wrap around readIndex here

  ADCXStr = Xtotal >> 3; // Total divided by numReadings (and convert to string)
```

```
  ADCYStr = Ytotal >> 3;
  //

  // If pen has changed state then output pen command
  // Pen UP signal is HIGH (relay open, interanl pull up resistor), pen DOWN is LOW (relay closed to ground)
  if (PenState != OldPenState) {
    (PenState == HIGH) ? (CmdStr = "PU;") : (CmdStr = "PD;");
    OldPenState = PenState; // Remember pen state
    Serial.println(CmdStr);
  }

  #ifdef __dummy_data
   // dummy data for now, plot a circle
    float t = (float)millis() / 1000.0;
    float arg = TWO_PI_F * t;
    ADCXStr = int(511.5*cos(arg)+511.5);
    ADCYStr = int(511.5*sin(arg)+511.5);
  #endif

  #ifdef __serial_plotter
    int Pen;
    (PenState == HIGH) ? Pen = 10 : Pen = 0;  // View the Pen up/down signal
    CmdStr = "1023 0 " + ADCXStr + " " + ADCYStr + " " + Pen; // 1023 0 prevents autoscaling
  #else
    // 'Plot Absolute' Ex: PA1023,250;
    CmdStr = "PA" + ADCXStr + "," + ADCYStr + ";";
  #endif

  Serial.println(CmdStr);

  // If switch went HIGH, output final commands, stop data stream, the plot is completed, wait here until RESET
  if (digitalRead(SwitchPin) == HIGH) {
    Serial.println("PU;");
    Serial.println("SP;");
    Serial.println("IN;");

    while (true) {  // wait here for RESET
      // flash 'idle' LED to indicate need for reset
      digitalWrite(IdleLED, HIGH);
      delay(200);
      digitalWrite(IdleLED, LOW);
      delay(300);
    }
  }
  //

  /* Wait here for remainder of loop time.
   * Full scale plot swing is 1023 counts in 1.667 seconds (5 volt full swing / 3.0 volts/second).
   * So, no more than one plot command every 1.630ms is necessary (1.667/1023). ~614 samples/second.
```

```
   */
  while (micros() - LoopTime < 1630) {;} // wait here for remainder of loop time

  // Short (0.250Âµs) output pulse to test loop timing.
  PORTB |= B1; PORTB ^= B1; // digitalWrite(TestPin, HIGH); digitalWrite(TestPin, LOW);
}  // end loop
```

Analog X-Y Pen Plotter Output To HP-GL Translator

Part List:

project:Tektronix 2232 X-Y Plotter to HPGL Translator file:Signal Conditioning Board - Rev H.fzz			
Reference	**Quantity**	**Part Number**	**Description**
C1,C2	2	USR1E100MDD1TP	Aluminum Electrolytic Capacitors - Leaded 10uF 25V 85c 4x7 20% 1.5LS
C3,C4,C5,C6,C7,C8	6	TAP104K035SRW	Tantalum Capacitors - Solid Leaded 35volts 0.1uF 10%
C9,C10	2	BFC237042682	CAP FILM 6800PF 5% 250VDC RADIAL
D1	1	1N5817	DIODE SCHOTTKY 20V 1A DO41
J1	1	68001-410HLF	CONN HEADER 10POS .100 STR TIN
LED	1	B4301H1/5	T1 3/4, 2 Lead, Red/Green LED with Clear Lens Cap Mounting Clip
R1	1	RN55D9101FB14	Metal Film Resistors - Through Hole 1/8watt 9.1Kohms 1% 100ppm
R2	1	RN55D1002FB14	Metal Film Resistors - Through Hole 1/8watt 10Kohms 1% 100ppm
R3	1	OF201JE	Carbon Composition Resistors 1/2W 200 ohms 5%
S1	1	TS02CBE	Slide Switches Switch Slide Spst Pc Mnt
U1	1	ICL7662CPA+	IC REG SWTCHD CAP INV RATIO 8DIP
U2,U3	2	AMP03GPZ	IC OPAMP DIFFERENTIAL 3MHZ 8DIP
U4	1	LT1460GIZ-2.5#PBF	IC VREF SERIES 2.5V TO92-3
U5	1	LT1460GCZ-5#PBF	IC VREF SERIES 5V TO92-3
Off board parts			
	Quantity	**Part Number**	**Description**
	1	15388100	CONN CIC FFC RCPT 10POS 2.54MM
	1	30-9506-99	CABLE DB9 FEMALE/MALE 6'
	1	SWI6-9-N-P5	AC/DC WALL MOUNT ADAPTER 9V 6W

Cost:

Arduino UNO	$25
Arduino Proto Shield (needed for the headers only. I spun my own board)	$8
Parts (25)	$57
Board by Aiser (qty 3)	$34 (€31)
Total	**$124 (€114)**

References:

- Atmel ATmega328/P 8-bit AVR Microcontroller Datasheet.
- Arduino Programming Nootebook by Brian W. Evans.
- Arduino Uno Rev3 Schematic
- Tektronix 2232 Digital Storage Oscilloscope User Manual, 070-7066-01
- Tektronix 2232 Digital Storage Oscilloscope Service Manual, 070-7067-01
- The HP-GL/2 and HP RTL Reference Guide, A Handbook for Program Developers, 5961-3526