LabView:
Temperature and Pressure Logger
Manual

Florian Heringa

July 23, 2021

Contents

1 Introduction

In this manual the "Temperature and Pressure Monitor" VI is explained. This VI was built to read out temperature and pressure values of the ARPES system at Science Park IOP. These values are shown on screen and can be logged live into a .csv file. The VI is also capable of controlling the temperature controller via serial connection to turn on the heating element and apply temperature ramps.

1.1 Specifications

1.1.1 Hardware

The VI reads temperature from a Lakeshore 332 Temperature Controller, communication VIs and drivers are supplied by NI.
The pressure readout is done from a Granville-Phillips 330 Vacuum Gauge Controller. Since our particular model only supplies data out on an analog output, an ATMEGA microcontroller was chosen to act as an ADC. Specifically the Adafruit ItsyBitsy 32u4 - 5V. This microcontroller board has an ATMEGA32u4 microcontroller with a 10 bit ADC on board. The output voltage of the Vacuum Gauge controllers is 0-10V and goes slightly over 10V in the case that pressures are too high. Therefore, to use the ADC with an internal voltage reference of 2.56V (which is much more stable than if it were referenced to the 5V supply through USB), the incoming signal needs to be attenuated. This is done by using a voltage divider per channel.

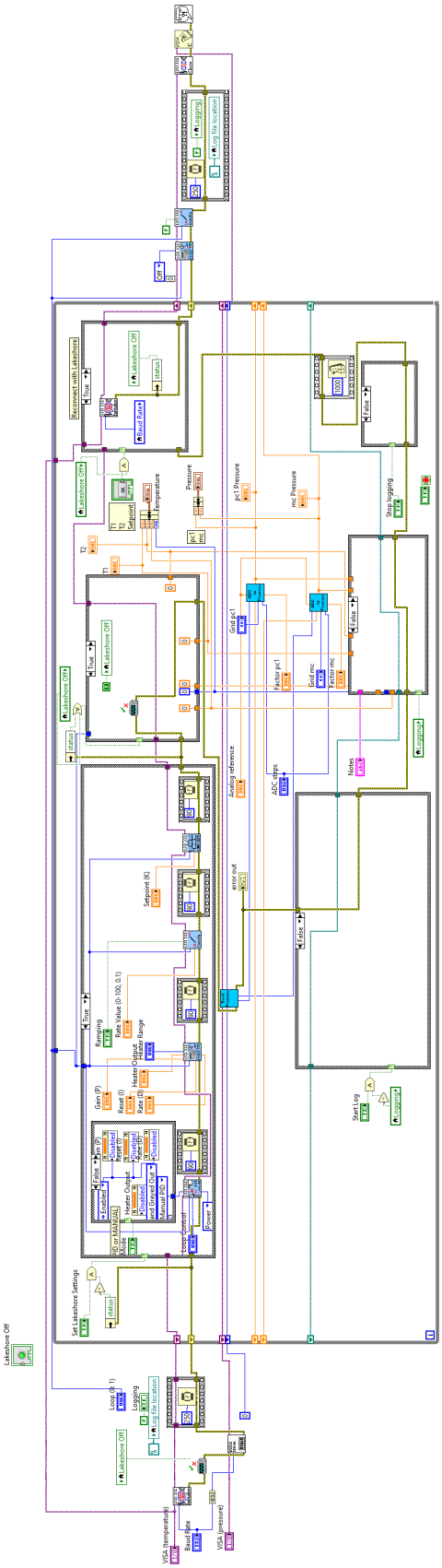
1.1.2 Software

The main control loop takes a bit more than half a second to complete without any additional delay methods. This is changed to exactly 1 second, as to have a sampling frequency of (almost exactly) 1Hz. The microcontroller sends out it's signal with a frequency of 2.5Hz, this is to ensure that there is always data available at the serial port when LabView tries to receive data.

1.1.3 Arduino code

```
// Read two Granville-Phillips 350 Vacuum Gauge Controllers
// The channels that are read are from the Ion gauges' analog out.
// This code was written for use with an ADAfruit Itsy-Bitsy (ATMEGA32u4), 5V
// It receives two analog signals on A0 and A1 from the Vacuum gauges.
// These signals are converted by the on-board (10 bit) ADC and sent through
// a serial connection to a LabView program which handles further data processing.

#define PC1_PIN A0 // Analog0
#define MC_PIN A1 // Analog1
#define NUM_INTEGRATION 50

// For storing adc values
int pc1_adc_val = 0;
int mc_adc_val = 0;
int n;

unsigned long int t1, t2;

// Byte arrays for sending data over serial
byte * pc1;
byte * mc;
byte startBytes[2] = {264, 264};

void setup()
{
  Serial.begin(9600);
  // Set analog reference to internal 2.56V reference
  // This is the internal voltage for the ATMEGA32u4
  analogReference(INTERNAL);
  // Cause why not
```

```
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop()
{
  // Store time for loop delay purposes
  t1 = millis();

  // Reset variables for integration
  pc1_adc_val = 0;
  mc_adc_val = 0;

  // Integrate over a few ADC values to
  for (n = 0; n < NUM_INTEGRATION; n++)
  {
    pc1_adc_val += analogRead(PC1_PIN);
    mc_adc_val += analogRead(MC_PIN);
  }

  // Divide by number of integration points
  pc1_adc_val /= NUM_INTEGRATION;
  mc_adc_val /= NUM_INTEGRATION;

  // Prepare data for serial transmission
  pc1 = (byte *) &pc1_adc_val;
  mc = (byte *) &mc_adc_val;

  // Write to serial port and blink LED
  digitalWrite(LED_BUILTIN, HIGH);
  Serial.write(0b11111111);
  Serial.write(0b11111111);
  Serial.write(pc1, 2);
  Serial.write(mc, 2);
  delay(10);
  digitalWrite(LED_BUILTIN, LOW);

  // Delay such that writing frequency to serial port
  // is 2.5 Hz to ensure no sync issues occur.
  // The labview program reads with afrequency of 1 Hz.
  while (millis() - t1 < 400)
  {
    continue;
  }
}
```


Lakeshore Off
Set Lakeshore Settings
Reconnect with Lakeshore
Lakeshore Off
Pressure
Temperature
pc1 Pressure
mc Pressure
Stop logging
Gain (P)
Reset (I)
Rate (D)
Heater Output
Heater Output Range
Setpoint (K)
Ramping
Ramp Value (0-100, 0.1)
Analog reference
ADC steps
Notes
Start Log
Loop (0-1)
Logging
log file location
VISA (temperature)
VISA (pressure)
Baud Rate

2 Schematic

2.1 LabView

The previous page shows the LabView block diagram of the rev.87 version (25/11/2019). The following explanation will follow the error cluster through the schematic.
The VI starts with opening a connection with the temperature controller and the Arduino. Ashort delay has been built in right after to ensure the connection has finished setting up. The first thing that is done inside the loop is setting the control type. Depending on the "Mode" switch on the front panel the control mode is set to PID or manual. The three following VI calls set the parameters for the heater (temperature, range and PID values), ramping and the setpoint. Then two VIs are used for reading the actual values of the setpoint and the heater output (these are different from the set values when ramping). Finally, as a last step for the temperature controller, the following two VIs read the measured temperature values.
Then the Arduino serial port is read. This is done with another VI which waits for two 0b11111111 bytes to occur at the serial port, after this signal it extracts exactly four bytes and interprets them as integers (these are the 0-1023 DAC values that come from the Arduino). These integers are converted to an actual voltage by using

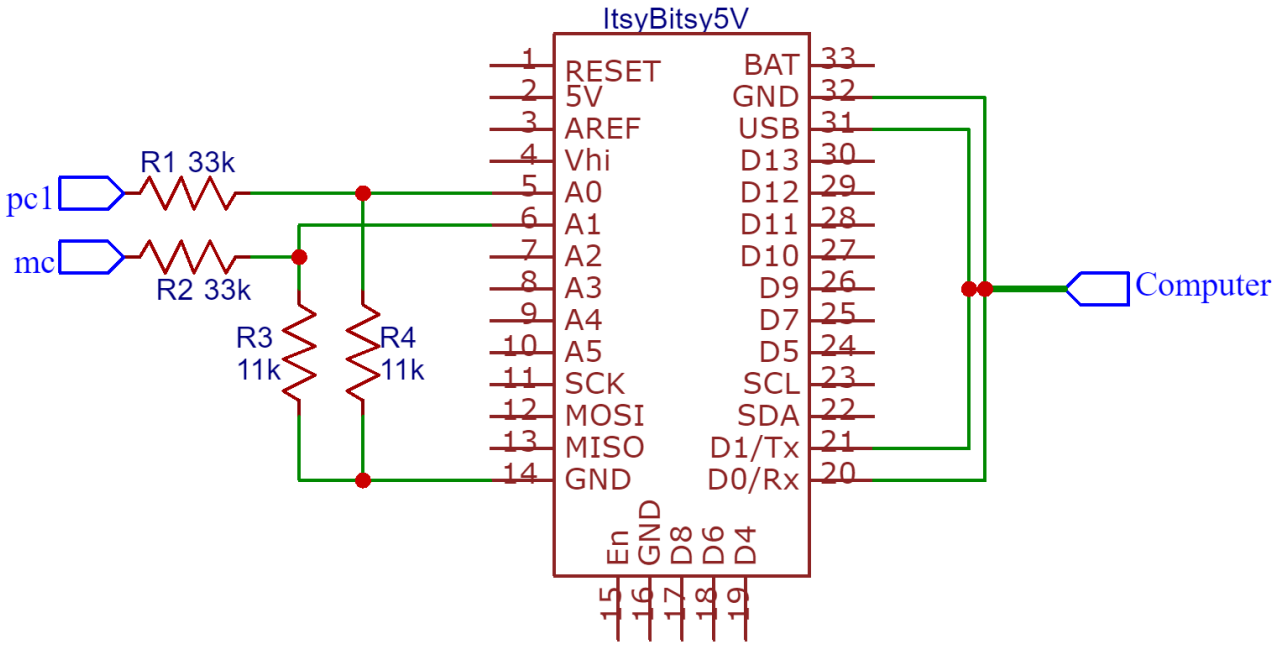
$$V_{\text{in}} = \frac{\text{ADC}_{\text{in}} * V_{\text{max}}}{1024}$$

where V_{max} is the internal reference voltage of the Arduino, which is 2.56V. Then this value is multiplied by 4 to account for the attenuation of the voltage dividers used for signal conditioning (see next section). Finally the calibration graph in the Vacuum gauge controllers is used to convert the voltage value into a pressure reading, this gives for the total pressure expression

$$P = 10^{4 \cdot \frac{\text{ADC}_{\text{in}} * V_{\text{max}}}{1024} - GC} \tag{1}$$

where GC is an integer from 10 to 12 depending on the grid current of the vacuum gauge.
When the user presses the "Start logging" button, the VI first creates a new comma separated file with corresponding headers in the chosen location. All relevant values are stored in this file while logging is active.
The final action in the loop is a short delay, which ensures that all data values are spaced by 1 second.
Then, after pressing the "Stop" button the VI turns off the heating elements, stops any ramps that are active and stops logging. Finally, serial communication is closed.
It is very important to always close the VI with the "Stop" button, because otherwise the heater element might not turn off properly.

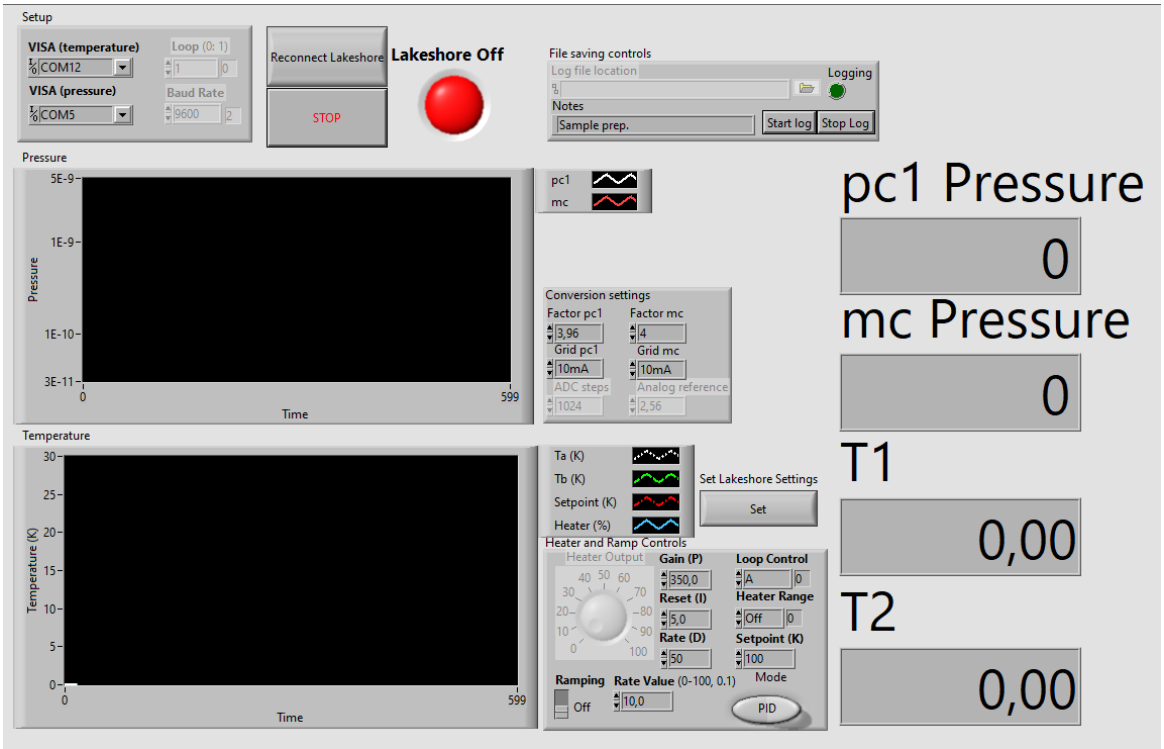
2.2 Hardware



The above schematic shows the connections to the microcontroller. Both analog signals are connected to 33k/11k voltage dividers which divide the incoming signal by 4. The pc1 signal is connected to A0 with a yellow wire,

and the mc signal is connected to A1 with a green wire. The connections are done with 3.5mm mono jacks, which are grounded on the microcontroller board (not shown in schematic).

3 Front Panel



To start the montoring VI, just press the "Run" button as usual for LabView VIs. This will start the VI and immediately begin displaying data in the graphs and on the numerical indicators. The two graphs shown are for pressure and temperature. The pressure graph displays the pressure in pc1 and mc, this is done for the last 600 samples, this corresponds to a time of 10 minutes. Similarly, the temperature graph shows temperatures for the cryostat and the sample holder for the same amount of samples and time.
The conversion setting panel shows the controls for converting the ADC value from the Arduino into a pressure reading as described in Formula 1. The "factor" control is dependent on the voltage divider connected between the Arduino and the analog output of the Vacuum Gauge controller. In theory this value should be exactly 4, but because of resistor tolerance this is a bit different. The "Grid" setting is the setting which controls the GC parameter in Formula 1, this value can be set on the front panel of the Vacuum Gauge controller. Finally, the "Analog reference" control is equal to the internal reference voltage on the Arduino. This can be measured on the "ARef" pin on the board.
In the top right the controls for saving a log file are shown. To start a new log, click the "Start log" button. This opens a file dialog where you can select a path and filename for the logfile. While logging, the green LED is lit and the "Log file location" indicator shows the path where the log file is being stored. It is also possible to enter notes here to indicate what was going on during the log file at specific times. These notes are also put into the log file at 1Hz. The log file is continuously streamed to the file so in case of an error the log file is unharmed. Finally, to stop the log click on the "Stop Log" button.
The last section of the front panel controls is the "Heater and Ramp Controls" section. Here you can set up a ramp and choose the PID setting or the raw output of the heater.

3.0.1 Setting up a ramp

1. Turn off "Ramping"
2. Set the temperature setpoint to the current temperature.
3. Set the "Rate Value", which is in K per minute.
4. Set the temperature setpoint to the desired temperature.
5. Turn the heater to the desired power.

6. Turn on "Ramping"

The heater can be configured in two modes. In "Manual" mode the heater simply power the heating element at the requested power and output percentage. In PID mode the heater will heat according to the chosen PID values.

3.0.2 Disconnecting the Temperature controller

During operation it became apparent that the temperature controller had some negative effects on the resolution of the measurement. Therefore a disconnect procedure was implemented. The temperature controller can be turned off without issue and the VI will keep logging pressure information. As soon as the temperature controller is turned on again the connection can be re-established by clicking the "Roconnect Lakeshore" button on the front panel.

4 Troubleshooting

General advice

If anything malfunctions, check the error cluster for help (scroll down on the front panel). In case of unclear issues, just close the VI, unplug the temperature controller and Arduino. Then plug everything back in, and turn on the VI. Try to make sure that you plug the hardware into the same USB port as before, or check the COM ports they are connected to.

4.1 Software

The VI is not running.

Check the "run" arrow key, if it displays a broken arrow something is wrong in the schematic. Use the LabView error finder to try and solve the issue.

The VI is not graphing the temperature or pressure.

This is probably a hardware issue, scroll down on the front panel and check the error cluster. If there are any connection errors restart the VI.

The software is not logging.

Did you enter a valid path for the file to be saved? Check the file location if there is a file there. If there is, the logging was probably interrupted because of connection issues with the hardware. If there is no log file present, then the path you entered might be invalid.

4.2 Hardware

The VI can't connect to....

Turn off the VI, temperature controller and Arduino. Then turn everything on again. Also check if the COM ports are set correctly. By default the temperature controller should be on COM6, and the pressure readout on COM9.

The heater element did not turn off.

Always make sure to close the VI with the "Stop" button on the front panel, and not the small LabView "Abort Execution" button. This just terminates all connections to external hardware without any communication.

The Arduino light is flashing very quickly

The exact cause is yet unknown, just unplug the USB cable and restart the VI.