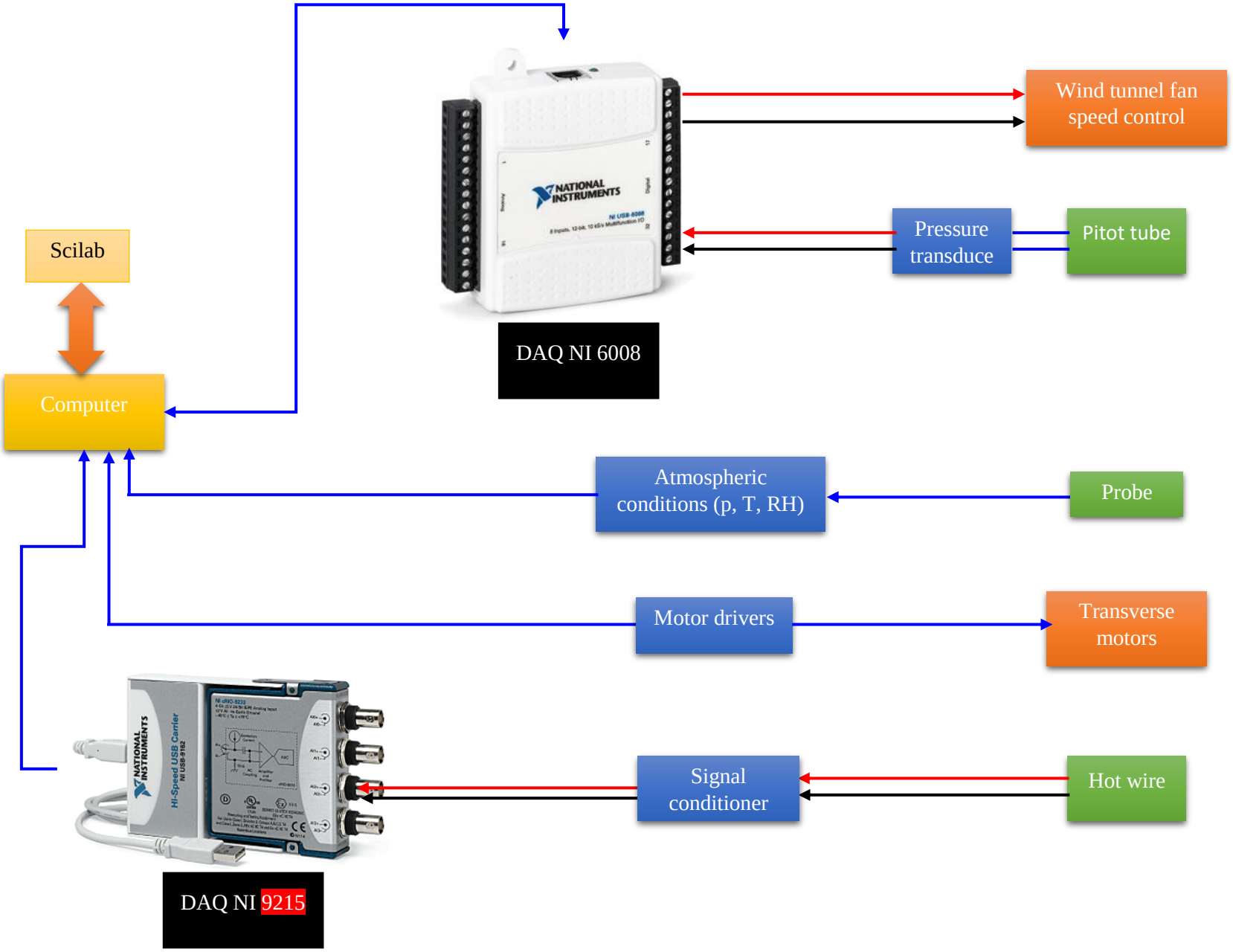
Scilab
Computer
DAQ NI 6008
Wind tunnel fan speed control
Pressure transduce
Pitot tube
Atmospheric conditions (p, T, RH)
Probe
Motor drivers
Transverse motors
Signal conditioner
Hot wire
DAQ NI 9215

The atmospheric conditions (p, T and RH) were measured using Comet H7331 device. The specs of the device are listed in table 1. The device was connected to the PC via a USB connection. The Modbus protocol was used to transfer data from the apparatus to Scilab. The Modbus module can be loaded on Scilab from the module manager – ATOMS. A simple Scilab code then can be elaborated to get the data from the Comet H7331 registers:

Unit	hPa, mBar	PSI	mmHg	inHg	inH2O	oz/in^2	kPa
Range	600	8.7	450	17.72	240.9	139.2	60
	1100	15.95	825.1	32.48	441.6	255.3	110
Accuracy							
T=23°C (T=73,4°F)	±1.3	±0.02	±1.0	±0.04	±0.5	±0.3	±0.13
0≤T≤40°C (32≤T≤104°F)	±1.5	±0.02	±1.1	±0.04	±0.6	±0.3	±0.15
Else	±2.0	±0.03	±1.5	±0.06	±0.8	±0.5	±0.20

```
function [T, RH, P]=SensorAtm(Time, Frequency)
   //Open atmospheric conditions modbus connection.
  //modbus_newRtu(device, baud, parity, data_bits, stop_bits)
   mbc = modbus_newRtu("COM6",9600,'N',8,2)
   modbus_connect(mbc)
   modbus_setSlave(mbc,1);
   modbus_setTimeout(mbc, -1)
   //Get data from the registers.
  //modbus_readInRegs(mbc, address, number of input registers to read)
   [T,e]= modbus_readInRegs(mbc,48,1)
   [RH,e]= modbus_readInRegs(mbc,49,1)
   [P,e]= modbus_readInRegs(mbc,51,1)

   //The data are received with no decimal points.
   T=T/10; RH=RH/10; DP=DP/10; P=P/100

   //Close atmospheric conditions modbus connection.
   modbus_close(mbc)
   modbus_free(mbc)
endfunction
```

A Pitot-static-tube was installed to get the mean flow velocity. The pressure difference across the tube legs was measured using an LSI differential pressure transducer (range: 0 – 1600 Pa). The pressure transducer signal was conditioned by an NI DAQ 6008 module before sending to PC via a USB cable. The data were read by Scilab through the NIDAQ module which can be loaded onto Scilab from ATOMS. The Scilab code used to read the pressure difference is shown below:

```
function [dP]=SensordP(Time, Frequency)
   //Open differential pressure transducer connection.
   [ taskAI, status ] = DAQ_CreateTask ("")
   //DAQ_CreateAIVoltageChan (TaskHandle taskHandle, const char physicalChannel[], const char nameToAssignToChannel[],
int32 terminalConfig, float64 minVal, float64 maxVal, int32 units, const char customScaleName[])
   DAQ_CreateAIVoltageChan( taskAI, "Dev2/ai0", DAQ("Val_Diff"),-5,5)
   //DAQ_CfgSampClkTiming (TaskHandle taskHandle, const char source[], float64 rate, int32 activeEdge, int32 sampleMode,
uInt64 sampsPerChanToAcquire)
   DAQ_CfgSampClkTiming (taskAI,"", Frequency, DAQ("Val_Rising"), DAQ("Val_ContSamps"), ceil(Time*Frequency))
   DAQ_StartTask ( taskAI )
   //DAQ_ReadAnalogF64 (TaskHandle taskHandle, int32 numSampsPerChan, float64 timeout, bool32 fillMode, float64 readArray[],
uInt32 arraySizeInSamps, int32 *sampsPerChanRead, bool32 *reserved)
   dP=DAQ_ReadAnalogF64 ( taskAI,ceil(Time*Frequency),-1,DAQ("Val_GroupByChannel"),ceil(Time*Frequency))
   dP=(dP-2.5898663)/(5.2164903-2.5898663)*1600      //Calibration
   DAQ_StopTask ( taskAI )
   DAQ_ClearTask ( taskAI )
endfunction

dPTime=10                  //Sample time (s)
dPFrequency=10000          //Sampling frequency (Hz)
```

The instantaneous velocity was measured using a Dantec hotwire anemometer. The signal of the hotwire was conditioned by a Dantec adapter then an NI DAQ 9215 module. The final voltage was read by Scilab using a code similar to that used to read the differential pressure transducer:

```
function [HotWireV]=SensorHW(Time, Frequency)
   [ taskAI, status ] = DAQ_CreateTask ("")
   DAQ_CreateAIVoltageChan( taskAI, "Dev3/ai0", DAQ("Val_Cfg_Default"),-10,10)
   DAQ_CfgSampClkTiming (taskAI,"", Frequency, DAQ("Val_Rising"), DAQ("Val_ContSamps"), ceil(Time*Frequency))
   DAQ_StartTask ( taskAI )
   HotWireV=DAQ_ReadAnalogF64 ( taskAI,ceil(Time*Frequency),-1,DAQ("Val_GroupByChannel"),ceil(Time*Frequency))
   DAQ_StopTask ( taskAI )
   DAQ_ClearTask ( taskAI )
endfunction

HWTime=10                  //Sample time (s)
HWFrequency=20000          //Sampling frequency (Hz)
```

The fan speed was controlled through an output voltage signal produced by the NI DAQ 6008 module under request of Scilab. The code is detailed below:

```
function MoveFanTo(Volt)
   //Open fan motor microcontroller connection.
   [ taskAO, status ] = DAQ_CreateTask ("")
   DAQ_CreateAOVoltageChan( taskAO, "Dev2/ao0",0,5)
   DAQ_StartTask ( taskAO )

   [a,b]=DAQ_WriteAnalogF64 ( taskAO,1,-1,DAQ("Val_GroupByScanNumber"),Volt)

   DAQ_StopTask ( taskAO )
   DAQ_ClearTask ( taskAO )
endfunction
```

The minimum volt (0) runs the fan at 0.56 Hz while the maximum volt (5) runs it at 49.6 Hz. The fan speed ranges between 0 (corresponding to 0 Hz) and 1450 rpm (corresponding to 50 Hz).

The traverse stepper motors were also controlled by Scilab, through Velmex motor controllers. This time a serial communication toolbox was loaded from ATOMS onto Scilab. The code used to control the two motors is shown in the following lines:

```
//M1 for vertical motion (mm)
//M2 for horizontal motion (mm)

//Control motor "1"
function M1(Distance)
   ScrewPitch=2.1                          //Distance per revolution (mm)
   Rev=Distance/ScrewPitch                 //Total number of revolutions
   SPR=400                                 //Number of steps per revolution
   Steps=-1*Rev*SPR                        //Total number of steps
   Speed=1000                              //Steps/second
   Time=abs(Steps/Speed)                   //Motion time (s)
   str = ['F C setM1M4,getM1M,(S1M' string(Speed) ',),(I1M' string(Steps) ',),R'];
   h=openserial(5,"9600,n,8,1")
   writeserial(h,str)
   closeserial(h)
   sleep(Time*1000+1000)
endfunction

//Control motor "2"
function M2(Distance)
   ScrewPitch=2.1                          //Distance per revolution (mm)
   Rev=Distance/ScrewPitch                 //Total number of revolutions
   SPR=400                                 //Number of steps per revolution
   Steps=Rev*SPR                           //Total number of steps
   Speed=400                               //Steps/second
   Time=abs(Steps/Speed)                   //Motion time (s)
   str = ['F C setM1M4,getM1M,(S2M' string(Speed) ',),(I2M' string(Steps) ',),R'];
   h=openserial(5,"9600,n,8,1")
   writeserial(h,str)
   closeserial(h)
   sleep(Time*1000+1000)
endfunction
```