Weather Station for the ROVOR Telescope

Evan Hansen

A senior thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Bachelor of Science

J. Ward Moody, Advisor

Department of Physics and Astronomy

Brigham Young University

August 2011

ABSTRACT

Weather Station for the ROVOR Telescope

Evan Hansen
Department of Physics and Astronomy
Bachelor of Science

All telescopes face vulnerability to adverse weather conditions during operations. This is particularly true of remote and robotic telescopes, such as ROVOR, which do not have an on-site operator to monitor the weather. To protect ROVOR, a weather station and monitoring program has been developed and constructed to provide a constant watch over the site. The afore mentioned weather station has been in operation for over a year at the time of this writing.

Keywords: ROVOR, Weather Station

ACKNOWLEDGMENTS

I would like to thank Dr. J. Ward Moody for his help during this project. And BYU department of Physics and Astronomy for their financial support of my research efforts.

Contents

Chapter 1

Introduction

1.1 Telescope - ROVOR

The Remote Observatory for Variable Object Research (ROVOR) is a 16" RC Optical telescope located in the west desert of Utah near the city of Delta. It is mounted on a Paramount ME Robotic Telescope System and is housed in a specially designed Lifferth Dome. Currently controlled by the Sky using Orchestrate, ROVOR is operated remotely from the Brigham Young University (BYU) campus. Further information about ROVOR can be found in the paper Remote Observatory for Variable Object Research: ROVOR (Moody 2010).

1.2 Vulnerability of ROVOR Telescope

Located approximately two hours by car from BYU campus and operated remotely the ROVOR telescope is particular vulnerable to sudden changes in weather. Not having an on-site operator there exists a need to be able to constantly monitor the weather conditions around the observatory. High winds can damage the dome if it is not closed quickly enough as well as carry objects and debris into the enclosure. Water, both rain and dew, can damage instruments and wiring and dirty

the mirrors and the other optical surfaces of the telescope.

1.3 Early Work on Weather Station

The first person to work on the weather station was Daniel Wilcox. He oversaw the procurement of the hardware and the initial work on both the sensor board and the software to run the weather station. After graduation he discontinued his work leaving the weather station incomplete. At this point I was put in charge of finishing the project and overseeing its set-up and instillation. My work was done over a period of several months starting in the spring 2008 and concluded in the fall of 2008. The instillation of the weather station at the site in Delta was conducted during the spring of 2009 by Richard Pearson and Cameron Pace.

Chapter 2

Hardware: Construction and Set-up

The weather station's hardware is made up of the six instruments used for meteorological measurement and the sensor interface board which processes the raw information from the instruments and transfers it to one of the observatory's computer.

2.1 Sensor Interface Board

2.1.1 Construction

The sensor interface board is constructed from two power supplies, one 5V and the other 12V, two op amps, a block of resistors, an input block, a 9 pin serial port, a PICkit 2 connection, and a PIC18F4423 enhanced flash microcontroller. The board is housed in a custom made steel box to protect it from the elements and is located in the observatory computer building known as the Outhouse.

2.1.2 Input Block, Stand-alone Pins and External Connections

The input block is provided as the means to connect the sensors to the interface board. Both the 5V and 12V power supply outputs are connected directly to the input block. As another means of connecting the sensors to the board a stand-alone pin exists for each of the input block ports and is wired directly to it.

For power, the use of any standard computer-style power cord will suffice.

2.1.3 Serial Communication

The board communicates over a 115200 baud serial (RS-232 style) interface. Only two of the nine pins are used: pin 5, which is ground, and pin 3, which transmits the signal. The PIC18F4423 sends a stream of raw measurements over the serial cable. The text for each measurement is shown in the form of an abbreviation followed by a number separated by an equals sign. The abbreviations are Wd, Ws, T, H, P, Rg, and Si (they stand for wind direction, wind speed, temperature, humidity, pressure, rain gauge, and solar intensity, respectively). The numbers fall between 0 and 4095, where 0 means 0 volts and 4095 means 5 volts.

2.2 Instrument Configuration

2.2.1 Instruments

As previously stated there are six individual sensor instruments that make up the weather station. Each of the instruments were manufactured by Texas Electronics. The six instruments are temperature/humidity, solar radiation, wind speed, wind direction, rainfall, and barometric pressure sensors (with product numbers TTH-315, SP-Lite, TV-114, TD-104-5D, TR-525USW-HT, and TB-2012M, respectively). These six instruments are located on a pole on the Northwest corner

of the enclosure and are connected to the sensor interface board via a underground PVC pipe that leads to the base of the Outhouse.

2.2.2 Wiring

Each of the sensors has a number of wires that must be connected to the correct ports in the input block on the sensor interface board. The correct placement for each wire will be described below and can be referenced in the diagram of the input block in the appendix.

The temperature/humidity sensor has four wires: red, black, white, and brown. Red connects to 5V, black to ground, white to "Temperature," and brown to "Humidity." The solar radiation sensor has two wires: clear and black. Clear connects to "Solar" and black to ground. The wind speed sensor has two wires: white and green. The two wires connect to 2.5V and "Wind Speed," it doesn't matter to which port each wire is connected to.

The wind direction sensor has three wires: red, black, and brown. Red connects to 5V, black to ground and brown to "Wind Direction." The rainfall sensor has three wires: clear, black and non-insulated. Clear connects to 5V, black to "Rain Gauge" and the non-insulated wire to ground. The barometric pressure sensor has three wires: red, black and white. Red connects to 12V, black to ground and white to "Pressure."

Chapter 3

Software: Data Collection and Analysis

Both of the following programs are simple in both design and purpose. Together they are meant to supply all the necessary meteorological information to safely run the ROVOR observatory. The code for both programs can be found in the appendix.

3.1 NOAA Listener Program

The National Oceanic and Atmospheric Administration (NOAA) listener computer program downloads an XML file from the NOAA server once every fifteen minutes. It produces a digest of meteorological information for the area surrounding Delta, UT and places it in a file labeled *noaa_report.txt*. Each time the program updates the information in the file is changed as opposed to be added to. And as currently implemented the program does not produce a log file.

The downloaded file is "$http://www.weather.gov/data/current\_obs/KU24.xml$".

3.2 Sensor Watcher Program

Intended to run in the background of the computer connected to the weather station, as currently implemented the program listens on COM1. It runs continuously whenever the computer is on, updating and changing the content file, *cur_weather.txt* every ten minutes. This file is found in the same directory as the program itself.

A second file $cur\_weather\_log < today's date > .txt$ is also produced. <Today's date> is of the form "Wed June 1 2011." Also found in the same directory as the program itself, this file is a log that appends the same information as the *cur_weather.txt* every ten minutes.

Chapter 4

Results and Conclusion

The weather station at the ROVOR telescope has so far performed exceptionally well. For over a year and one half the station has monitored the conditions around the telescope and have help every member of the ROVOR research group to optimize the use of the telescope and to prevent damage from occurring to its instruments. Any and all telescopes, whether operated on-site or remotely, would benefit from a weather station and monitoring program such as the one described here. And with an increasing number of remote and robotic telescope being operated around the world a small investment in a complete weather station will protect the much larger investment made in the telescopes themselves.

4.1 Possible Future Improvements

Though the weather station has been very successful in its purpose so far there is one addition to the project that would improve what it is already doing. This would be to create a software program that would we able to read the text file produced by the sensor watcher program and react to any sudden changes in weather around the site. If the information in the text file were to fall outside of certain parameters this new program would be able to stop the script running the telescope and start

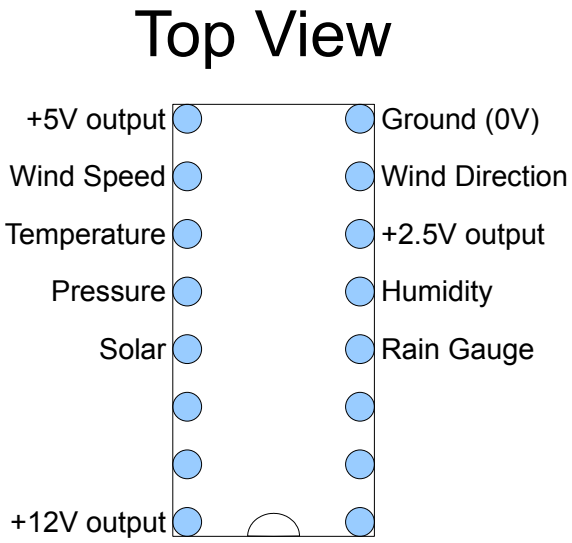
a preset script that would shut down the telescope and close the dome, thus protecting the telescope from harm. This program could also be set to send a warning message to the group members by e-mail or any other method, keeping them appraised of any severe changes at the site.

I would highly recommend that such a program be written for the continued success of the ROVOR telescope.

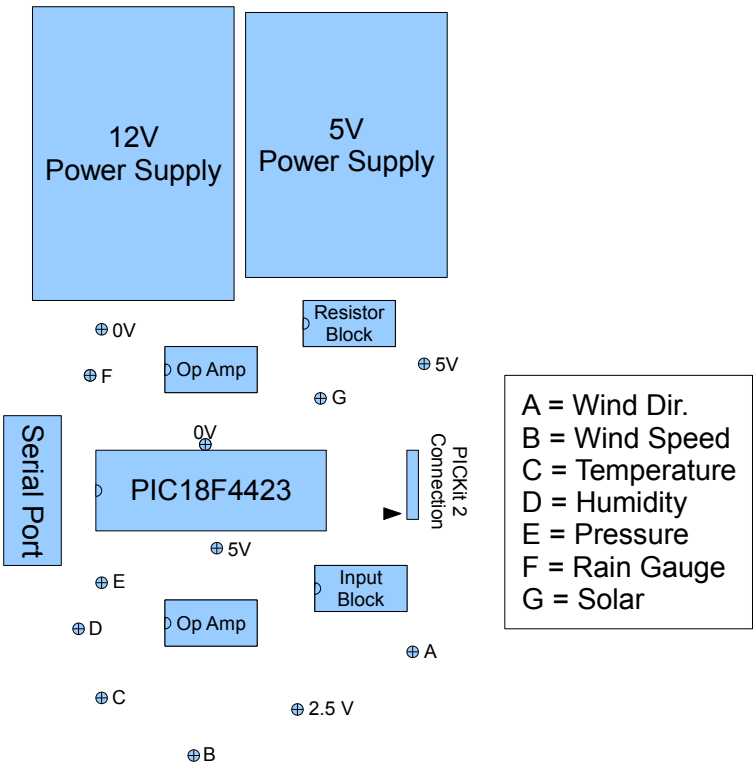
Appendix A

Specifications

A.1 Input Block and Sensor Interface Board



block diagram.pdf



interface board drawing.pdf

Appendix B

Software Codes

B.1 NOAA Listener Program Code

```
// noaa_listener.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"

#include <urlmon.h>

#include <iostream>
#include <fstream>
#include <string>
using namespace std;

#include <winerror.h>
```

```
string find_attr(string whole_file, string tag)
{
string start_tag = "<" + tag + ">";
string end_tag = "</" + tag + ">";

string::size_type s = whole_file.find( start_tag );
string::size_type e = whole_file.find( end_tag );

if(s == string::npos)
return "";
if(e == string::npos)
return "";
if(e < s+start_tag.length())
return "";

return whole_file.substr(s+start_tag.length(), e-s-start_tag.length());
}

int _tmain(int argc, _TCHAR* argv[])
{
while(1)
{
// Initialize the COM library
CoInitialize(NULL);
```

```
const unsigned int bufsize = 4096;
TCHAR buf[bufsize];
buf[0] = 0;
// Note that "http://www.weather.gov/data/current_obs/KU24.xml" is the URL for the
//  current observation from Delta, UT
const TCHAR *url = _T("http://www.weather.gov/data/current_obs/KU24.xml");

bool succeed_download = false;
while(!succeed_download)
{
HRESULT r = URLDownloadToCacheFile(NULL, url, buf, bufsize, 0, NULL);
if(r == S_OK)
{
succeed_download = true;
}
else
{
cerr << "Error trying to download web page
                \"http://www.weather.gov/data/current_obs/KU24.xml\"." << endl;
Sleep(100);
cerr << "Retrying..." << endl;
}
}
// Uninitialize the COM library
CoUninitialize();
```

```
ifstream in_file(buf);
if(!in_file.is_open())
{
cerr << "Could not open cached downloaded web page." << endl;
exit(-2);
}
string whole_file;
while(!in_file.eof())
{
string cur_line;
getline( in_file, cur_line );
//cout << cur_line << endl;
whole_file += cur_line;
}
in_file.close();

// Now it's time to take that XML file and exract what
// we want from it.
ofstream cur_report("noaa_report.txt");
cur_report << find_attr(whole_file, "observation_time") << endl;
cur_report << "Temperature (F) = " << find_attr(whole_file, "temp_f") << endl;
cur_report << "Humidity (%) = " << find_attr(whole_file, "relative_humidity") << endl;
cur_report << "Weather = " << find_attr(whole_file, "weather") << endl;
cur_report << "Wind speed (mph) = " << find_attr(whole_file, "wind_mph") << endl;
cur_report << "Wind direction (degrees) =
```

```
         " << find_attr(whole_file, "wind_degrees") << endl;
cur_report << "Visibility (miles) = " << find_attr(whole_file, "visibility_mi") << endl;
cur_report << "Pressure (in) = " << find_attr(whole_file, "pressure_in") << endl;
cur_report.close();

// Sleep for 15 minutes.  Note that 1000*60*15 does not overflow.
// I don't have to use MsgWaitForMultipleObjects because I've
// already uninitialized the COM library.
Sleep( 1000 * 60 * 15 );
}

return 0;
}
```

B.2 Example of noaa_report.txt File

```
Last Updated on Nov 1, 9:46 am MDT

Temperature (F) = 38

Humidity (%) = 77

Weather = Clear

Wind speed (mph) = NA

Wind direction (degrees) = NA

Visibility (miles) = 50.00

Pressure (in) = NA
```

B.3 Sensor Watcher Program Code

```
<?xml version="1.0" encoding="Windows-1252"?>

<VisualStudioProject

ProjectType="Visual C++"

Version="8.00"

Name="sensor_watcher"

 ProjectGUID="{88F0C1A3-070A-4321-A54F-09B76DE8CB94}"

RootNamespace="sensor_watcher"

Keyword="Win32Proj"

>

<Platforms>

<Platform

Name="Win32"

/>

</Platforms>

<ToolFiles>

</ToolFiles>

<Configurations>

<Configuration

Name="Debug|Win32"

 OutputDirectory="$(SolutionDir)$(ConfigurationName)"

 IntermediateDirectory="$(ConfigurationName)"

ConfigurationType="1"

CharacterSet="1"

>
```

```
<Tool
Name="VCPreBuildEventTool"
/>
<Tool
Name="VCCustomBuildTool"
/>
<Tool
Name="VCXMLDataGeneratorTool"
/>
<Tool
Name="VCWebServiceProxyGeneratorTool"
/>
<Tool
Name="VCMIDLTool"
/>
<Tool
Name="VCCLCompilerTool"
Optimization="0"
 PreprocessorDefinitions="WIN32;_DEBUG;_CONSOLE"
MinimalRebuild="true"
BasicRuntimeChecks="3"
RuntimeLibrary="3"
UsePrecompiledHeader="2"
WarningLevel="3"
Detect64BitPortabilityProblems="true"
```

```
DebugInformationFormat="4"
/>
<Tool
Name="VCManagedResourceCompilerTool"
/>
<Tool
Name="VCResourceCompilerTool"
/>
<Tool
Name="VCPreLinkEventTool"
/>
<Tool
Name="VCLinkerTool"
AdditionalDependencies="kernel32.lib $(NoInherit)"
LinkIncremental="2"
GenerateDebugInformation="true"
SubSystem="1"
TargetMachine="1"
/>
<Tool
Name="VCALinkTool"
/>
<Tool
Name="VCManifestTool"
/>
```

```
<Tool
Name="VCXDCMakeTool"
/>
<Tool
Name="VCBscMakeTool"
/>
<Tool
Name="VCFxCopTool"
/>
<Tool
Name="VCAppVerifierTool"
/>
<Tool
Name="VCWebDeploymentTool"
/>
<Tool
Name="VCPostBuildEventTool"
/>
</Configuration>
<Configuration
Name="Release|Win32"
 OutputDirectory="$(SolutionDir)$(ConfigurationName)"
 IntermediateDirectory="$(ConfigurationName)"
ConfigurationType="1"
CharacterSet="1"
```

```
WholeProgramOptimization="1"
>
<Tool
Name="VCPreBuildEventTool"
/>
<Tool
Name="VCCustomBuildTool"
/>
<Tool
Name="VCXMLDataGeneratorTool"
/>
<Tool
Name="VCWebServiceProxyGeneratorTool"
/>
<Tool
Name="VCMIDLTool"
/>
<Tool
Name="VCCLCompilerTool"
 PreprocessorDefinitions="WIN32;NDEBUG;_CONSOLE"
RuntimeLibrary="2"
UsePrecompiledHeader="2"
WarningLevel="3"
Detect64BitPortabilityProblems="true"
DebugInformationFormat="3"
```

```
/>
<Tool
Name="VCManagedResourceCompilerTool"
/>
<Tool
Name="VCResourceCompilerTool"
/>
<Tool
Name="VCPreLinkEventTool"
/>
<Tool
Name="VCLinkerTool"
AdditionalDependencies="kernel32.lib $(NoInherit)"
LinkIncremental="1"
GenerateDebugInformation="true"
SubSystem="1"
OptimizeReferences="2"
EnableCOMDATFolding="2"
TargetMachine="1"
/>
<Tool
Name="VCALinkTool"
/>
<Tool
Name="VCManifestTool"
```

```
/>
<Tool
Name="VCXDCMakeTool"
/>
<Tool
Name="VCBscMakeTool"
/>
<Tool
Name="VCFxCopTool"
/>
<Tool
Name="VCAppVerifierTool"
/>
<Tool
Name="VCWebDeploymentTool"
/>
<Tool
Name="VCPostBuildEventTool"
/>
</Configuration>
</Configurations>
<References>
</References>
<Files>
<Filter
```

```
Name="Source Files"
 Filter="cpp;c;cc;cxx;def;odl;idl;hpj;bat;asm;asmx"
 UniqueIdentifier="{4FC737F1-C7A5-4376-A066-2A32D752A2FF}"
>
<File
RelativePath=".\DataCollector.cpp"
>
</File>
<File
RelativePath=".\Listener.cpp"
>
</File>
<File
RelativePath=".\sensor_watcher.cpp"
>
</File>
<File
RelativePath=".\stdafx.cpp"
>
<FileConfiguration
Name="Debug|Win32"
>
<Tool
Name="VCCLCompilerTool"
UsePrecompiledHeader="1"
```

```
/>
</FileConfiguration>
<FileConfiguration
Name="Release|Win32"
>
<Tool
Name="VCCLCompilerTool"
UsePrecompiledHeader="1"
/>
</FileConfiguration>
</File>
</Filter>
<Filter
Name="Header Files"
Filter="h;hpp;hxx;hm;inl;inc;xsd"
 UniqueIdentifier="{93995380-89BD-4b04-88EB-625FBE52EBFB}"
>
<File
RelativePath=".\DataCollector.h"
>
</File>
<File
RelativePath=".\Listener.h"
>
</File>
```

```
<File
RelativePath=".\stdafx.h"
>
</File>
</Filter>
<Filter
Name="Resource Files"
 Filter="rc;ico;cur;bmp;dlg;rc2;rct;bin;rgs;
                gif;jpg;jpeg;jpe;resx;tiff;tif;png;wav"
 UniqueIdentifier="{67DA6AB6-F800-4c08-8B7A-83BB121AAD01}"
>
</Filter>
<File
RelativePath=".\ReadMe.txt"
>
</File>
</Files>
<Globals>
</Globals>
</VisualStudioProject>
```

B.4 Example of cur_weather.txt File

```
Tue Aug 28 18:47:45 2010


Current wind dir: 0.0107943 degrees.

Current wind speed: 4 mph.

Current temperature: 60.15 degrees Farenheit.

Current humidity: 65.101%.

Current pressure: 32.898".

Rainfall in the last ten minutes: 0".

Solar intensity: 155.084 W/m^2.
```

Bibliography

Moody, J. W. 2010, *tfa.cfht.hawaii.edu/papers/moody_tfa_paper.pdf*