DiSEqC Low Cost Antenna Tracker

Christian Monstein

The goal of this project was to design and manufacture an antenna tracker that could not only track the Sun, but also other objects in the sky (Moon, satellites, etc.). Our main priority was to keep the cost as low as possible to ensure everyone could build such a unit. The key components (and most expensive ones) are two DiSEqC [1] compatible satellite rotors and an Arduino Micro [2] on a printed circuit board (PCB). The first DiSEqC-channel is controlling azimuth or hour angle, depending on Python-application, while the second is controlling elevation or declination, again depending on Python application. A single Arduino Micro controls the positioning of both.

Figures 1 and 2 show the completed unit and table 1 lists the specifications. Figures 3 and 4 and table 2 show the schematic, basic PCB layout and parts list. Figure 5 shows how two azimuth rotors can be mounted to provide azimuth and elevation.

For some latitudes it might be sufficient to track the Sun with a single azimuth-only rotor close by the path of geostationary satellites, assuming the beam angle of the antenna covers at least 20°. In such a special case, the satellite-rotor can be used like in television satellite applications. Figure 6 shows the solution at Glasgow university in UK. Figures 7 and 8 show additional applications, and figure 9 shows a laptop running the tracking program.

Fig. 1: Back panel of the DISEqC tracker with two F-connectors for satellite rotors, USB-control port and DC-input

Fig. 2: Front panel with power switch and LED

Table 1: Technical specifications of the DiSEqC controller Antenna Tracker

Attributes	Value
DiSEqC rotors	TechniSat, HH90, HH100 etc. or equivalent
DC supply	13 V - 18 V / 0.5 A min
Firmware/Application	Arduino C / Python 2.7 or higher
Signal type	DiSEqC 22 KHz
DiSEqC return	None
Angular deflection	Depending on rotor: +/- (62°…78°)
Angular resolution	1°
Current when idle	30 mA - 50 mA
Current when moving	200 mA - 350 mA

The cable connections are straightforward. We use a micro USB cable for the communication between the Arduino and the PC or notebook. The TV satellite cables with F-connectors are for communications between the controller and the satellite rotors. Any voltage between 13 and 18 volts works just fine. The higher the voltage the faster the drive velocity. For calculations of azimuth/elevation or hour angle/declination, a simple Python script is used which is executed every then and now, e.g. every 5 minutes to update the rotors position. Any other language can be used as only simple ASCII commands need to be sent to the rotors.

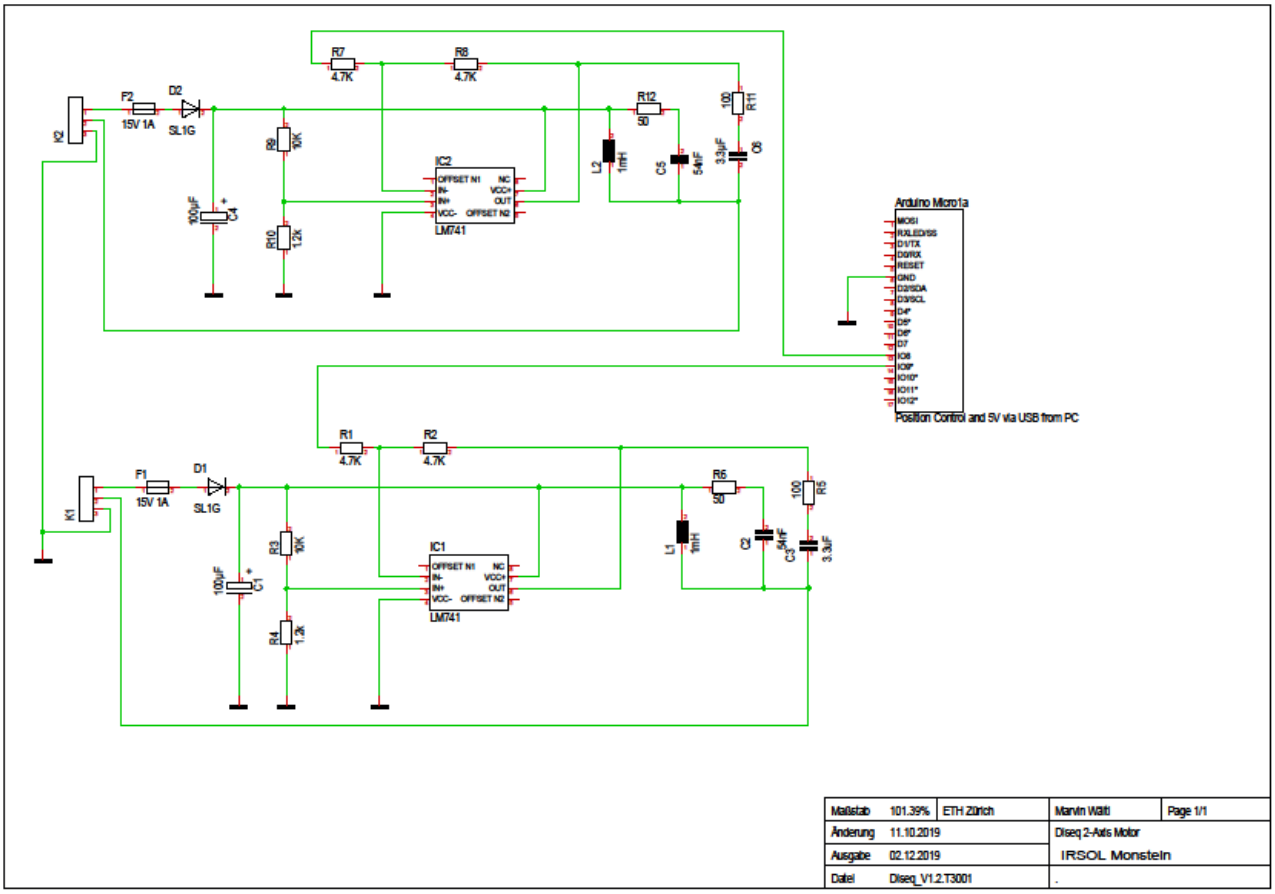

Fig. 3: Schematic of the dual channel rotor controller, based on the Arduino Micro. Processor is powered via the USB cable. The schematic and PCB layout are based on the TARGET 3001 PCB CAD program (link for downloading the file is provided later).

Fig. 4: Print circuit board at approximately full size. The dimensions are 100 mm x 53 mm, and the Arduino Micro is in the center. The large, black circular components are the coils L1 and L2.

Table 2: Parts list for one tracker unit with two channels

Qty	Component	Value	Size	Remark
1	PCB	Dual layer	100 x 53 mm	Beta Layout
1	Front plate	anodized	110 x 54 x 1.5	
1	Back plate	anodized	110 x 54 x 1.5	
4	R1, R2, R7, R8	4.7K	0603	
2	R5, R11	100	0603	
2	R6, R12	51	0603	
2	R4, R10	1.2k	0805	
1	Enclosure	Aluminum	165 x 110 x 55 mm	
1	Switch	1-pole	6mm	Front panel
1	Power adapter	18V 1.33A	With international adapters	external
1	USB cable	Black 1m		external
2	C2, C5	54nF	0805	
2	C3, C6	3.3uF	0805	
2	L1, L2	1mH	1210	
1	LED	Red	8mm	Front panel
1	Arduino	Micro1	Micro_Shield_Rev3	Soldered to PCB
2	C1, C4	47µF / 25V	SMD_R5X6_ELKO	
2	IC1, IC2	LM741	SO8	
2	D1, D2	SL1G	SOD123	Protection
2	F1, F2	15V 1A	USF1206	
1	DC-socket	Male 5.5/2.2mm	8mm	Back panel
2	F-connector	Female	9mm	Back panel
1	Fixing plate	PVC	112 mm x100 mm x 1 mm	PCB fixation
2	K1, K2	Stecker	SL-MTA/2.54/3POL	
2	R3, R9	10K	0603	
1	R_LED	2k2	Wired	Use shrinking hose

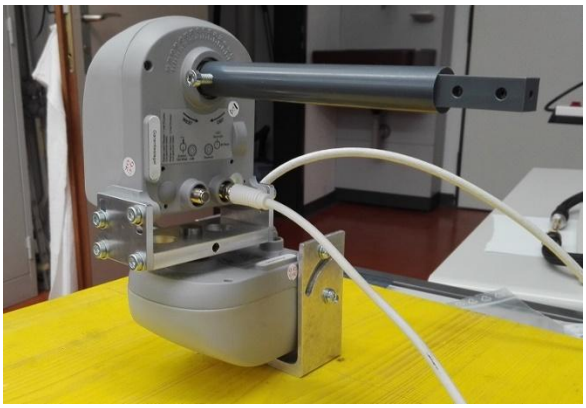

Fig. 5: How to stack two SAT-rotors for an Azimuth/Elevation drive system. The drive shafts were replaced by individual adapters to provide a two-axis system.

Fig. 6: How to install SAT-rotor for pseudo parallactic mount with tracking in local hour angle only at fixed declination. Declination is changed manually 2...4 times a year.

Fig. 7: S-band antenna with a satellite rotor on a tripod tracking the Sun

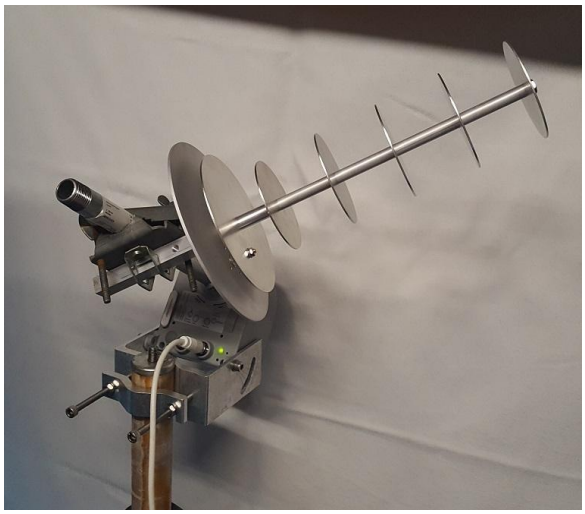

Fig. 8: Simple rotor with a shaft extension made from a piece of 1/2 inch water pipe. For demonstration a small L-band antenna has been attached. This configuration would also allow to point and track geostationary satellites.

Fig. 9: Control notebook with Windows 7 and the rotor controller box. A Python script is called every 5 minutes by the Splinterware System Scheduler program (ssfree.exe) and is commanding the rotor in figure 7 to the actual Sun position.

Software Installation

Required: Python 2.7 or higher, no guarantee for older versions

In case you need to improve or update Python script, I recommend Anaconda and Spyder [3]

1. Depending on your current configuration, install the following extra Python libraries:

    pip install serial to update type: pip install --upgrade serial

    pip install ephem to update type: pip install --upgrade ephem

    pip install orbital to update type: pip install --upgrade orbital

2. Install Arduino IDE or at least the driver for Arduino from here: https://www.arduino.cc/en/main/software

3. Install system scheduler from here:

    https://www.filecluster.com/downloads/System-Scheduler.html

    Note: In a LINUX system you can use crontab which is similar to the Windows ssfree.exe

4. Edit parameter-array MyLocation according to your latitude, longitude, altitude, pressure and temperature. For MaxRange check the specification of your satellite-rotor and for communication port check your device manager devmgmt.msc

If you are neither happy with PYTHON nor with the system scheduler ssfree.exe, you can always send commands to the rotor manually, based on a simple terminal program like PUTTY or WinSCP or HYPERTERMINAL or any other script which allows access to serial ports.

Table 3: Commanding the DiSEqC controller. Angles less than 1° are automatically round to an integer value.

Function	Command	Example
Set azimuth or hour angle	aziX	azi55 <ENTER>
Set elevation or declination	eleX	ele-12 <ENTER>
Set maximum deflection	maxX	Max65 <ENTER>
Get a response (draft firmware)	?	? <ENTER>
Get firmware version (current firmware)	-v	-v <ENTER>
Get help (current firmware)	-h	-h <ENTER>
Communication parameter	9600 Baud, 8N1 no handshake	

Installation, commissioning

Install the antenna pole in a vertical position as precise as possible, every error in tilt produces pointing error in hour-angle, declination, elevation or azimuth. Adjust the rotors azimuth in exactly north-south direction using a magnetic compass or even better find out when the Sun is in the meridian and adjust azimuth accordingly. Send command azi0 to the controller, such that the rotor and antenna are pointing to south. Then you may adjust rotor and antenna together in a way that the shadow of the front-dipole is exactly in the center of the antenna. Change Python script back into original version and run it. Depending on your location and depending on mechanical mounting method you may change the sign of hour-angle ha -> -ha

Example in Python for controlling azimuth/elevation

```
# -*- coding: utf-8 -*-
"""
This script tracks the Sun in azimuth/elevation mode
Carefully check parameter in MyLocation
It is sufficient to execute this script once every 4...5 minutes
Created on Thu Sep 14 20:13:50 2017
@author: Monstein
"""
# http://rhodesmill.org/pyephem/tutorial.html
# http://rhodesmill.org/pyephem/quick#other-observer-methods
#------------------------------------------------------------------------------
import ephem
import serial
import datetime
import math
import numpy as np
#------------------------------------------------------------------------------
MyLocation = ephem.Observer()
MyLocation.lon      = '8.7575' # east +°
MyLocation.lat      = '47.205833' # north +°
MyLocation.elev     = 414 # altitude in m asl
MyLocation.temp     = 20  # °C
MyLocation.pressure = 900 # mbar
MaxRange  = 75 # (+/- value) depends on your SAT-rotor type, check data sheet
MyComport = 'COM17' # check with device manager (C:\Windows\System32\devmgmt.msc)
```

```
#--------------------------------------------------------------------------
dt = datetime.datetime.now() # PC must run on UT or GPS-time
MyLocation.date = '{:4d}/{:02d}/{:02d} {:02d}:{:02d}:{:02d}'.format(
  dt.year,dt.month,dt.day,dt.hour,dt.minute,dt.second)
 #MyLocation.date = '2018/06/25 11:24:00' # example for testing at high noon
print 'Current date-time: ',MyLocation.date,' UT'
sun = ephem.Sun()
sun.compute(MyLocation)
azi = math.degrees(sun.az)
ele = math.degrees(sun.alt)
print("Sun data: Azimuth   =%6.2f  Elevation   =%6.2f" % (azi-180.0,ele))
lst = MyLocation.sidereal_time()
ha = (lst - sun.ra)/math.pi*180.0
dec = math.degrees(sun.dec)
print("Sun data: Hourangle =%6.2f  Declination =%6.2f" % (ha,dec))
myazi = azi - 180.0 # conversion 0° ... 360° -> +/- 180°
#--------------------------------------------------------------------------
if ((np.abs(myazi) < MaxRange) and (np.abs(ele) < MaxRange)):
   try:
      DiSEqC = serial.Serial(
          port     = MyComport,
          baudrate = 9600,
          bytesize = serial.EIGHTBITS,
          parity   = serial.PARITY_NONE,
          timeout  = 2)

      if (DiSEqC.isOpen()):
         print "Successfully connected to antenna tracker at: "+DiSEqC.portstr
         cmd = 'max{:6.2f}\r'.format(MaxRange)
         DiSEqC.write(cmd) # set MaxRange
         cmd = 'azi{:6.2f}\r'.format(myazi)
         DiSEqC.write(cmd) # set azimuth or hour angle
         cmd = 'ele{:6.2f}\r'.format(ele)
         DiSEqC.write(cmd) # set elevation or declination
         DiSEqC.close()
   except IOError:
      DiSEqC.close()
      print "Problem communication with tracker. Check COM-port and cables/connectors!"
else:
    print 'Sun out of rotor-range of +/-',MaxRange
#--------------------------------------------------------------------------
```

The original script can be downloaded for free from here:
http://www.e-callisto.org/Hardware/Diseqc/sunpos_AZI_ELE.py

And a similar script to control a satellite rotor set in hour-angle / declination can be downloaded for free from here:

http://www.e-callisto.org/Hardware/Diseqc/sunpos_HA_DEC.py

Design of the PCB in TARGET3001 can be downloaded from here (any students version of Target can be used to get all relevant docs out of Diseq_V1.1.T3001):

http://www.e-callisto.org/Hardware/Diseqc/Diseq_V1.1.T3001

Additional information about the controller can be downloaded from here:

http://www.e-callisto.org/Hardware/Diseqc/Doku%20Diseq.pdf

Access to the Arduino firmware is available here:
http://e-callisto.org/Hardware/Diseqc/Easy_Diseq/Easy_Diseq.ino

References:

[1] DiSEqC: https://en.wikipedia.org/wiki/DiSEqC
[2] Arduino: https://www.arduino.cc/
[3] Anaconda/Spyder: https://www.anaconda.com/

Christian Monstein is a native of Switzerland and lives in Freienbach. He obtained Electronics Engineer, B.S. degree at Konstanz University, Germany. Christian is a SARA member and is licensed as amateur radio operator, HB9SCT. He has 21 years of experience designing test systems in the telecommunications industry and is proficient in several programming languages including C++, IDL and PYTHON. He has worked at ETH-Zürich on the design of a noise transmitter as payload on a drone and is responsible for the hardware and software associated with the e-CALLISTO Project. He plays also the role of a coordinator of SetiLeague in Switzerland and he is also representing Switzerland within CRAF. He is a member of the ISWI steering committee at UN office for outer space affairs in Vienna (UNOOSA) and has just been nominated as member of ITU. Email contact: monstein(at)irsol.ch