

RPI YU5R Rotator

by Goran Stankovic dip.ing.el. – YT2FSG – Date: 07.04.2014.

email: goranstank@gmail.com



YU5R Rotator

192.168.1.16

YU5R
RESAVA VALLEY CONTEST CLUB

YU5R Rotator - Remote Control

Antenna 7MHz

Antenna 14MHz

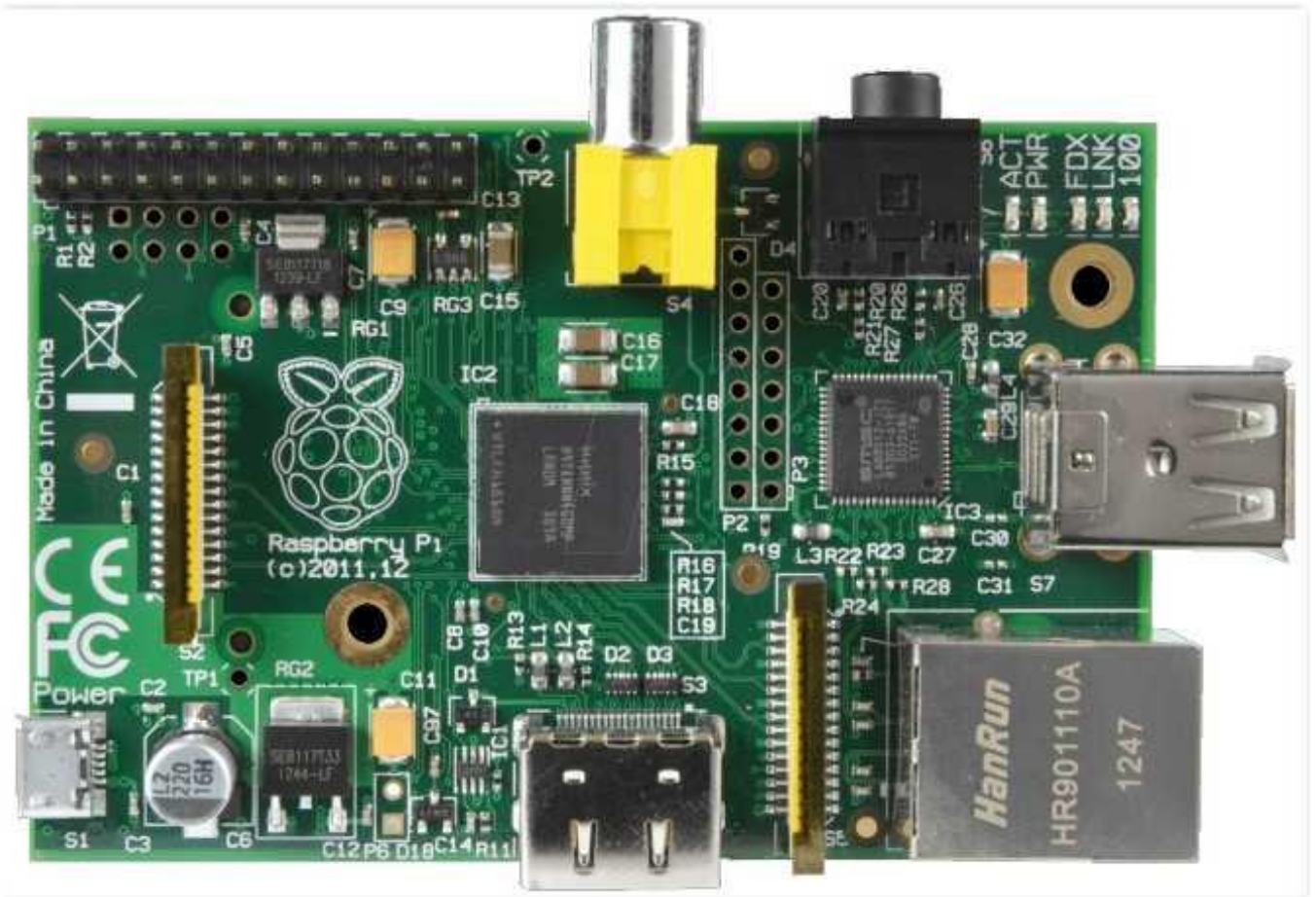
Antenna 21MHz

Antenna 28MHz

Refresh

Position: 150 (232)

Azimuth: GO



After installing a new Operating System on your Raspberry Pi's SD card you might notice there is a lot of disk space 'missing' on the card... By expanding the root partition of the card you can regain that 'missing' space!
This is the easiest way to do it:
open the Raspberry Pi's configuration screen (in the terminal window) by typing:

```
$ sudo raspi-config
Testing space at disk:
$ df -h
Testing space at memory:
$ free -h
```

Download and Install Apache and PHP:

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install apache2
$ sudo apt-get install php5
```

Download and Install Midnight Commander:

```
$ sudo apt-get install mc
```

By default the serial port on the Raspberry Pi is configured for console input/output. If you'd like to use the serial port in your own software, you must make the following changes.

First use nano to open `/boot/cmdline.txt` and edit it to disable boot info being sent to the port.

```
$ sudo nano /boot/cmdline.txt
```

The contents of the file should look a bit like this:

```
dwc_otg.lpm_enable=0 console=ttYAMA0,115200 kgdboc=ttYAMA0,115200 console=ttY1 root=/dev/mmcblk0p2
rootfstype=ext4 elevator=deadline rootwait
```

Remove the text highlighted in red, this will stop boot from being sent on startup. Press `Ctrl+O` to write the changes and `Ctrl+X` to exit nano. Now we need to edit `/etc/inittab` to disable console login on the serial port. Open this file with nano

```
$ sudo nano /etc/inittab
```

Look for this line:

```
T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100
```

And comment it out by adding # to the start of it. You can then save this file and exit nano like you did before. Now that you have made these changes we can reboot the Raspberry Pi.

```
$sudo reboot
```

Your serial port should now be ready for you to use in your own software.

The serial connection is owned by the **dialout** group so we need to add www-data to the dialout group to enable the PHP script to access the serial device.

```
$ sudo chmod a+rw /dev/ttyAMA0
```

```
$ chown -R www-data:www-data /var/www
```

```
$ chmod -R 755 /var/www
```

```
$ ls -l /dev/ttyAMA0
```

```
$ sudo usermod -a -G dialout www-data
```

Check that the dialout has been added to the group using the command groups **www-data**:

```
$ groups www-data
```

```
www-data:www-data dialout
```

Now you can reboot the Raspberry PI.

```
$sudo reboot
```

Now the UART works through the PHP script.

Download and Install Minicom:

```
$ sudo apt-get install minicom
```

```
$ minicom -b 9600 -o -D /dev/ttyAMA0
```

Exit with: Ctrl-A Q

Testing serial communication over Minicom.

For proper operation UART ports over PHP, make software for testing in gcc:

```
$ gcc uart test.c -o uart test
```

In file /etc/rc.local:

```
$ sudo nano /etc/rc.local
```

adding line:

```
sudo /home/pi/uart test
```

Download and Install WiringPi:

If you do not have GIT installed, then under any of the Debian releases (e.g. Raspbian), you can install it with:

```
$ sudo apt-get install git-core
```

If you get any errors here, make sure your Pi is up to date with the latest versions of Raspbian:

```
$sudo apt-get update
```

```
$sudo apt-get upgrade
```

To obtain WiringPi using GIT:

```
$git clone git://git.drogon.net/wiringPi
```

If you have already used the clone operation for the first time, then

```
$cd wiringPi
```

```
$git pull origin
```

Will fetch an updated version then you can re-run the build script below.

To build/install there is a new simplified script:

```
$cd wiringPi
```

```
$./build
```

The new build script will compile and install it all for you – it does use the sudo command at one point, so you may wish to inspect the script before running it.

Manual for Wiring Pi

```
$ man gpio
```

Testing Wiring Pi in the terminal window:

```
$ gpio -g read 17
$ gpio -g mode 17 out
$ gpio -g write 17 1
$ gpio -g write 17 0
```

Testing Wiring Pi in PHP:

```
$val = trim(@shell_exec("/usr/local/bin/gpio -g mode 17 out"));
$val = trim(@shell_exec("/usr/local/bin/gpio -g write 17 1"));
$val = trim(@shell_exec("/usr/local/bin/gpio -g write 17 0"));
```

Unpack files yu5r_rotator.tar to directory /var/www

```
$ tar xvf yu5r_rotator.tar
```

```
$ ls /var/www/
index.php          head.php          navi.php          start.php
open_port.php     open_port1.php   open_port2.php   open_port3.php
open_port4.php    php_serial.class.php antena.txt       azimuth.txt
read.me           yu5r_360.jpg    yu5r.ico         yu5r_1.jpg
```

For packing Files:

```
$ tar cvf name.tar /dir_name
```

You can connect various antenna rotator controllers such as the Yaesu GS-232, LVB Tracker <http://www.g6lvb.com/articles/lvbtracker/>, Fox Delta ST2 – Satelite Tracking Interface <http://www.foxdelta.com/products/st2-rs232.htm>, ...

or make one according to the scheme YU5R Antenna Rotator Controller.

You need 4 pcs antenna rotator controllers, one for each antenna: 7MHz, 14MHz, 21MHz, 28MHz.

Some antenna rotator can be connected over mini DIN, such as: YAESU G-800DX, YAESU G-1000DX, YAESU G-2800DX, some need the modification by adding the relays, such as: HY-GAIN HAM-IV, HY-GAIN HAM-V, HY-GAIN HDR-300A, YAESU G-450A, YAESU G-800SA, YAESU G-1000SA ...



Calibration Antenna Rotator Controller

After the unit is first programmed, it knows little about the voltages representing the rotator's maximum and minimum settings. You can set these either from your computer (using the serial connection) or from the front panel buttons.

Calibration from the serial connection

- (1) Connect to the unit using a terminal emulator at 9600 bps, 1 stop bit, no parity and no handshake.
- (2) Check for communication by typing the 'C' key followed by the enter key. Note that there is no echo of your keystrokes. Commands are also not case sensitive. The azimuth should be returned, although it will be the uncalibrated value.
- (3) Using either the rotator control box switches or the optional front panel switches, move the rotator to its counter-clockwise end stop (zero degrees), which should be due North or due South depending on your rotator.
- (4) Move the elevation to zero degrees.
- (5) To set the left-most azimuth, type these four keys in sequence: 'F', 'A', 'S', Enter. You will receive a response like: **Az Offset=0099**
- (6) To set zero degrees elevation, type the four keys in sequence: 'F', 'E', 'S', Enter. You will receive a response like: **EI Offset=0326**
- (7) Using either the rotator control box switches or the optional front panel switches, move the rotator clockwise 360 degrees for both (a) 360 degree maximum azimuth rotators and (b) 450 degree South CCW stop rotators. For 450 degree North CCW stop rotators, move the rotator clockwise to 450 degrees.
- (8) Move the elevation to either 90 degrees, or 180 degrees, depending on the end stop of your rotator.
- (9) To set the right-most azimuth for both (a) 360 degree maximum azimuth rotators and (b) 450 degree South CCW stop rotators, type these four keys in sequence: 'F', 'A', 'E', Enter. For 450 degree North CCW stop rotators, type the four keys in sequence: 'F', 'A', 'F', Enter. In either case, you will receive a response like: **Az mul=0.897**
- (10) For 90 degree elevation rotators only: to set 90 degrees elevation, type the four keys in sequence: 'F', 'E', 'N', Enter. For 180 degree elevation rotators only: to set 180 degrees elevation, type the four keys in sequence: 'F', 'E', 'E', Enter. In either case, you will receive a response like: **EI mul=1.234**
- (11) If you have a South CCW stopping rotator, you should key in the three key sequence 'F', 'S', Enter. For a North CCW stopping rotator (default), key in the three key sequence 'F', 'N', Enter.
- (12) To commit this to EEPROM, type the three keys in sequence: 'F', 'W', Enter.
- (13) When you reset or switch the unit on and off, the calibration parameters should be remembered.

Calibration from the front panel buttons

You must have an LCD connected for this to work. To enter calibration mode, hold down one of the four front panel buttons as the unit is switched on. After the welcome screen (and potentially the Bad EEPROM message), you then enter the calibration mode.

The four buttons (from LSB to MSB) are Left, Right, Down, Up. On the LCD, these are referred to as L, R, D, U.

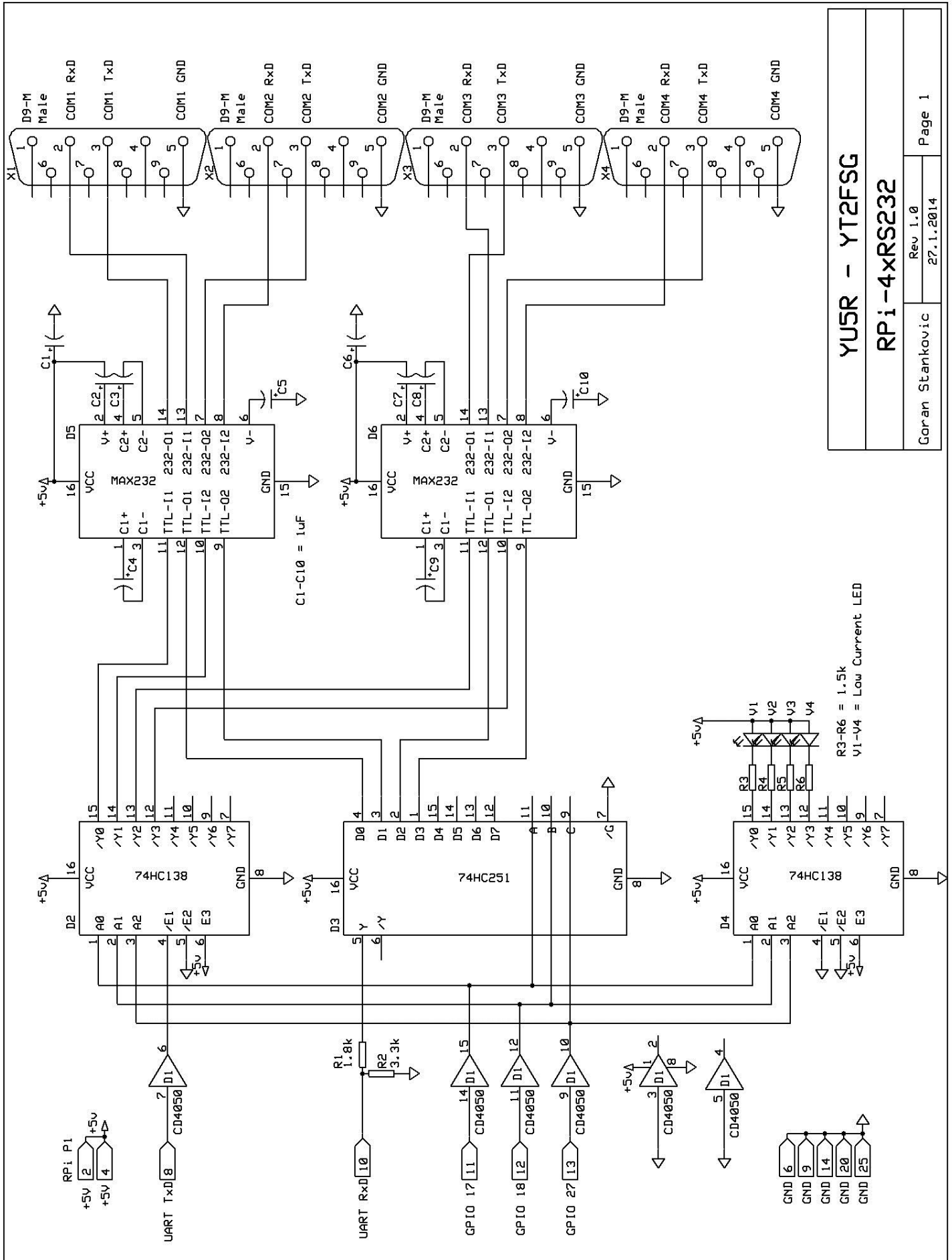
- (1) **Set Az=min & press D (U=esc)** Using the L button, move the rotator to its counter-clockwise stop. The R button also operates to move the rotator right, although normally would not be required at this point. Press D to accept, or U to abort the calibration completely.
- (2) **Set El=min & press L (R=esc)** Using the D button, move the rotator to the zero elevation stop. The U button also operates to move the rotator up, although normally would not be required at this point. Press L to accept, or R to abort the calibration completely.
- (3) **Set Az=max & press D (U=esc)** Using the R button, move the rotator to its clockwise stop. If you are using a 450 degree azimuth rotator with a South CCW stop, you should move the rotator to be at the South position 360 degrees clockwise of the CCW stop. The L button also operates to move the rotator left, although normally would not be required at this point. Press D to accept, or U to abort the calibration completely.
- (4) **Set El=max & press L (R=esc)** Using the U button, move the rotator to the maximum elevation stop (90 or 180 degrees). The D button also operates to move the rotator down, although normally would not be required at this point. Press L to accept, or R to abort the calibration completely.
- (5) **Az range: 450 U 360 D (R=esc)** For North CCW stop 450 degree rotators, press U. For 360 degree or South CCW stop rotators, press D. Press R to abort the calibration completely.
- (6) **EI range: 0-90 U 0-180 D (R=esc)** For elevation rotators with a maximum elevation of 90 degrees, press U. For 'flipping' elevation rotators, press D. Press R to abort the calibration completely.
- (7) **CCW stop North=U South=D (R=esc)** For North CCW stop rotators, press U, for South CCW stop rotators press D. Press R to abort the calibration completely.
- (8) **EE Write: U (R=esc)** To commit the settings to EEPROM, press U. Press R to abort the calibration completely.
- (9) **EEPROM write OK Push btn to exit** Push any front panel button to return to normal mode.

Quick configuration

If you know the calibration values for Az and El off and mul, you can set them directly using the following commands:

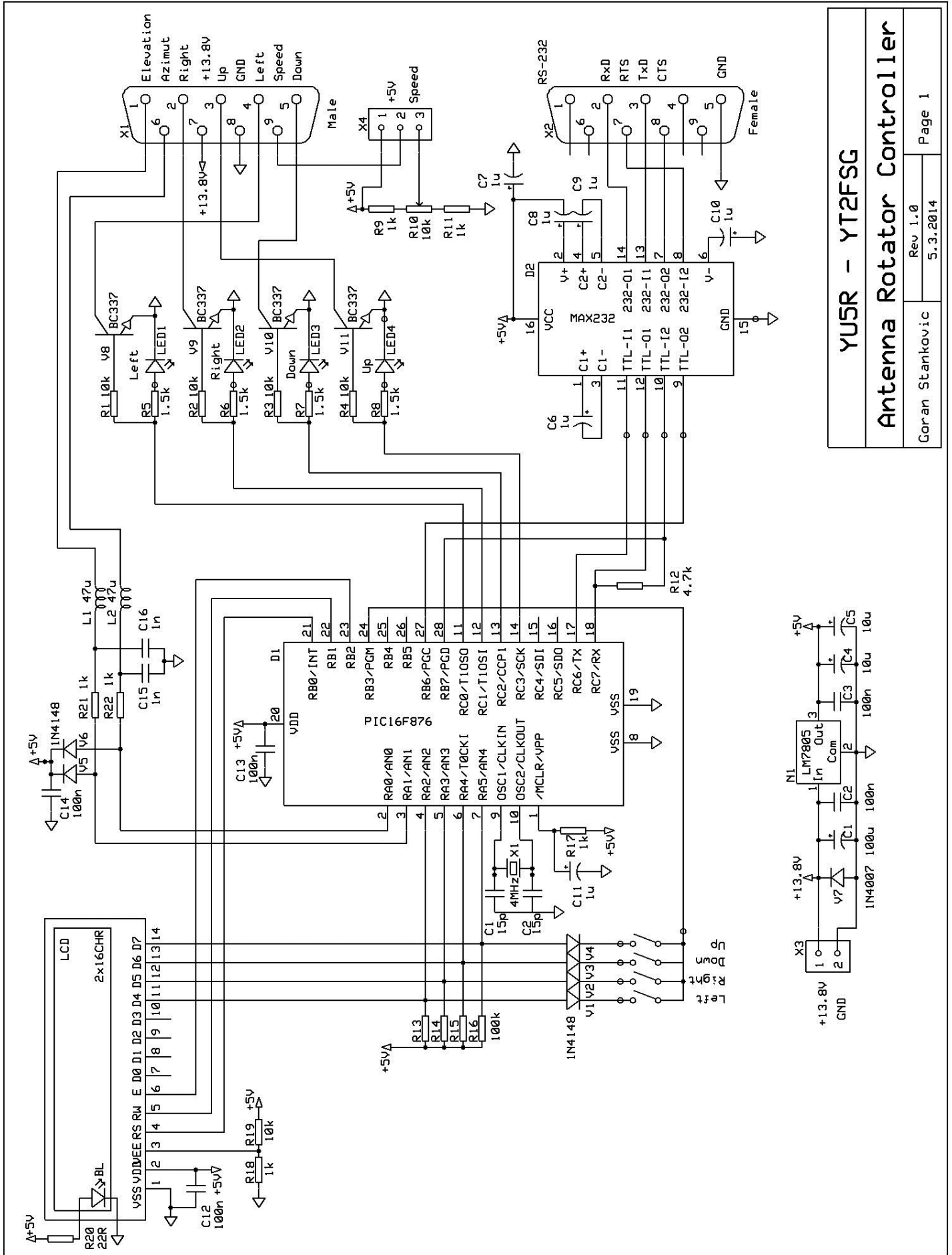
Az Off: FAOnnn
 Az Mul: FAMn.nn
 El Off: FEOnnn
 El Mul: FEMn.nn

Schematic Rpi-4xRS232



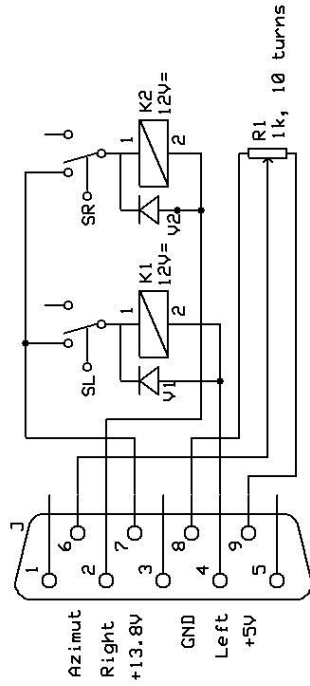
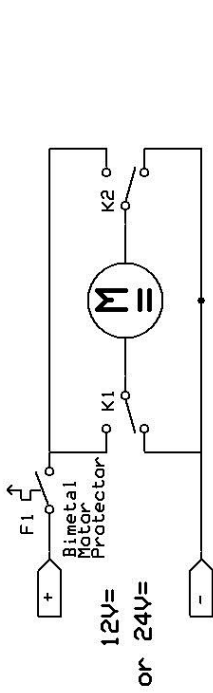
YU5R - YT2FSG
RPi-4xRS232

Schematic Antenna Rotator Controller



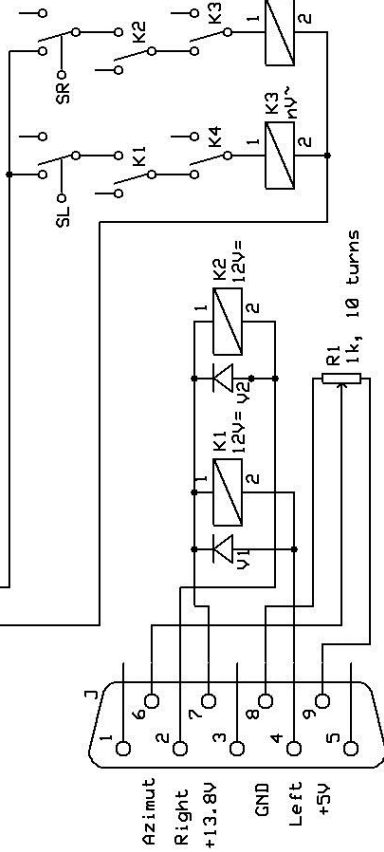
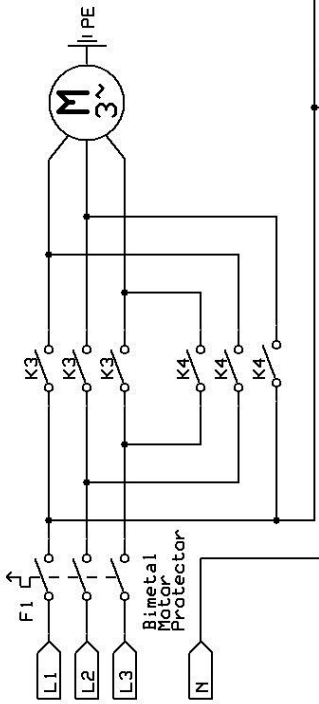
YU5R - YT2FSG	
Antenna Rotator Controller	
Goran Stankovic	Rev. 1.0
Page 1	
5.3.2014	

Homemade Antenna Rotator - Motor DC



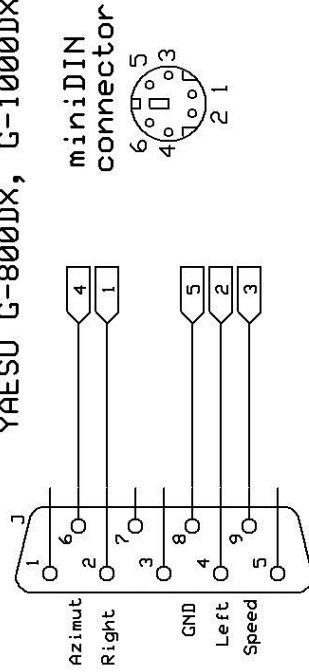
V1, V2 = 1N4007

Homemade Antenna Rotator - Motor AC 3~

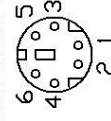


V1, V2 = 1N4007

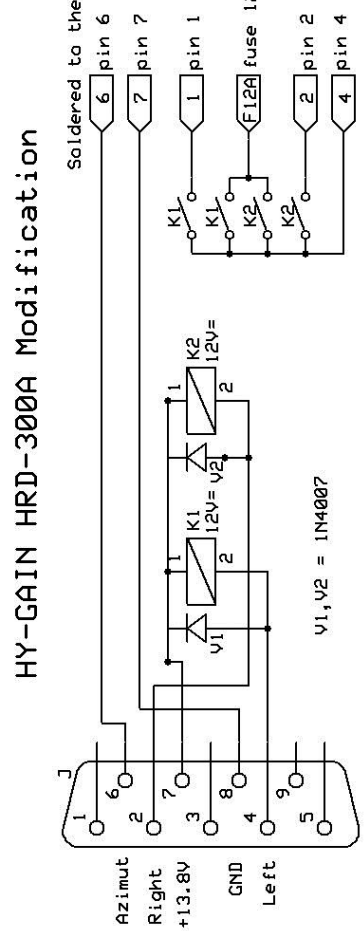
YAESU G-800DX, G-1000DX



miniDIN connector



Azimut
Right
GND
Left
Speed



V1, V2 = 1N4007

YU5R - YT2FSG Rotators

File: index.php

```

<html>
<head>
<meta http-equiv="cache-control" content="no-cache">
<title>YU5R Rotator</title>
<link rel="icon" href="yu5r.ico" type="image/ico" />
<frameset cols="220,*" marginwidth="0" marginheight="0" frameborder="0" border="0" borderwidth="0">
  <frameset rows="250,30,*" marginwidth="0" marginheight="0" frameborder="0" border="0" borderwidth="0">
    <frame name="head" src="head.php" scrolling=no>
    <frame name="info" src="open_port.php" marginwidth=20 scrolling=no>
    <frame name="navi" src="navi.php" marginwidth=20 scrolling=no>
  </frameset>
  <frame name="content" src="start.php" marginwidth=20>
</frameset>
</head>
<body bgcolor=#ffffff>
</body>
</html>

```

File: head.php

```

<html>
<body bgcolor=#dddeee>

<h2>YU5R Rotator - Remote Control</h2>
</body>
</html>

```

File: navi.php

```

<html>
<head>
<style type="text/css"><!--
#b_r {
background-color:red;
height:30px;
width:180px;
}
#b_w {
height:30px;
width:180px;
}
/-->
</style>
</head>

<body bgcolor=#ddeeee>
<?php
$filename = "antena.txt";
$handle = fopen($filename, "r");
$antena = fread($handle, filesize($filename));
fclose($handle);
?>

<table border="0" cellpadding="0" cellspacing="0">
  <tr valign="top">
    <td align="center">

<a target="info" href="open_port1.php" >
<?php
if ($antena=="1") { echo '<button id="b_r">';}
else { echo '<button id="b_w">';}
?>
Antenna 7MHz </button></a><br>

<a target="info" href="open_port2.php" >
<?php
if ($antena=="2") { echo '<button id="b_r">';}
else { echo '<button id="b_w">';}
?>
Antenna 14MHz </button></a><br>

```

```

<a target="info" href="open_port3.php" >
<?php
if ($antena=="3") { echo '<button id="b_r">';}
else { echo '<button id="b_w">';}
?>
Antenna 21MHz </button></a><br>

<a target="info" href="open_port4.php" >
<?php
if ($antena=="4") { echo '<button id="b_r">';}
else { echo '<button id="b_w">';}
?>
Antenna 28MHz </button></a><br>
<br>
<a target="content" href="start.php"><button id="b_w"> Refresh </button></a><br>

        </td>
    </tr>
</table>
</body>
</html>

```

File: open_port.php

```

<html>
<body bgcolor=#ddeeff>
<?php
$filename = "antena.txt";
$handle = fopen($filename, "r");
$antena = fread($handle, filesize($filename));
fclose($handle);
$val = trim(@shell_exec("/usr/local/bin/gpio -g mode 17 out"));
$val = trim(@shell_exec("/usr/local/bin/gpio -g mode 18 out"));
$val = trim(@shell_exec("/usr/local/bin/gpio -g mode 27 out"));
if ($antena=="1")
{
$val = trim(@shell_exec("/usr/local/bin/gpio -g write 17 0"));
$val = trim(@shell_exec("/usr/local/bin/gpio -g write 18 0"));
$val = trim(@shell_exec("/usr/local/bin/gpio -g write 27 0"));
echo "Antenna 7MHz";
}
elseif ($antena=="2")
{
$val = trim(@shell_exec("/usr/local/bin/gpio -g write 17 1"));
$val = trim(@shell_exec("/usr/local/bin/gpio -g write 18 0"));
$val = trim(@shell_exec("/usr/local/bin/gpio -g write 27 0"));
echo "Antenna 14MHz";
}
elseif ($antena=="3")
{
$val = trim(@shell_exec("/usr/local/bin/gpio -g write 17 0"));
$val = trim(@shell_exec("/usr/local/bin/gpio -g write 18 1"));
$val = trim(@shell_exec("/usr/local/bin/gpio -g write 27 0"));
echo "Antenna 21MHz";
}
elseif ($antena=="4")
{
$val = trim(@shell_exec("/usr/local/bin/gpio -g write 17 1"));
$val = trim(@shell_exec("/usr/local/bin/gpio -g write 18 1"));
$val = trim(@shell_exec("/usr/local/bin/gpio -g write 27 0"));
echo "Antenna 28MHz";
}
else
{
$val = trim(@shell_exec("/usr/local/bin/gpio -g write 17 0"));
$val = trim(@shell_exec("/usr/local/bin/gpio -g write 18 0"));
$val = trim(@shell_exec("/usr/local/bin/gpio -g write 27 0"));
echo "Antenna ?";
}

```

```

}
?>
</body>
</html>

```

File: open_port1.php

```

<html>
<body bgcolor=#ddeeef>
<?php
//echo "Antenna 7MHz";
$val = trim(@shell_exec("/usr/local/bin/gpio -g mode 17 out"));
$val = trim(@shell_exec("/usr/local/bin/gpio -g mode 18 out"));
$val = trim(@shell_exec("/usr/local/bin/gpio -g mode 27 out"));
$val = trim(@shell_exec("/usr/local/bin/gpio -g write 17 0"));
$val = trim(@shell_exec("/usr/local/bin/gpio -g write 18 0"));
$val = trim(@shell_exec("/usr/local/bin/gpio -g write 27 0"));
$filename = "antena.txt";
$handle = fopen($filename, "w");
fwrite($handle, "1");
fclose($handle);
echo '
<script type="text/javascript">
parent.navi.location.reload();
</script>';
?>
</body>
</html>

```

File: open_port2.php

```

<html>
<body bgcolor=#ddeeef>
<?php
//echo "Antenna 14MHz";
$val = trim(@shell_exec("/usr/local/bin/gpio -g mode 17 out"));
$val = trim(@shell_exec("/usr/local/bin/gpio -g mode 18 out"));
$val = trim(@shell_exec("/usr/local/bin/gpio -g mode 27 out"));
$val = trim(@shell_exec("/usr/local/bin/gpio -g write 17 1"));
$val = trim(@shell_exec("/usr/local/bin/gpio -g write 18 0"));
$val = trim(@shell_exec("/usr/local/bin/gpio -g write 27 0"));
$filename = "antena.txt";
$handle = fopen($filename, "w");
fwrite($handle, "2");
fclose($handle);
echo '
<script type="text/javascript">
parent.navi.location.reload();
</script>';
?>
</body>
</html>

```

File: open_port3.php

```

<html>
<body bgcolor=#ddeeef>
<?php
//echo "Antenna 21MHz";
$val = trim(@shell_exec("/usr/local/bin/gpio -g mode 17 out"));
$val = trim(@shell_exec("/usr/local/bin/gpio -g mode 18 out"));
$val = trim(@shell_exec("/usr/local/bin/gpio -g mode 27 out"));
$val = trim(@shell_exec("/usr/local/bin/gpio -g write 17 0"));
$val = trim(@shell_exec("/usr/local/bin/gpio -g write 18 1"));
$val = trim(@shell_exec("/usr/local/bin/gpio -g write 27 0"));
$filename = "antena.txt";
$handle = fopen($filename, "w");
fwrite($handle, "3");
fclose($handle);
echo '
<script type="text/javascript">
parent.navi.location.reload();
</script>';

```

```
?>
</body>
</html>
```

File: open_port4.php

```
<html>
<body bgcolor=#ddeeef>
<?php
//echo "Antenna 28MHz";
$val = trim(@shell_exec("/usr/local/bin/gpio -g mode 17 out"));
$val = trim(@shell_exec("/usr/local/bin/gpio -g mode 18 out"));
$val = trim(@shell_exec("/usr/local/bin/gpio -g mode 27 out"));
$val = trim(@shell_exec("/usr/local/bin/gpio -g write 17 1"));
$val = trim(@shell_exec("/usr/local/bin/gpio -g write 18 1"));
$val = trim(@shell_exec("/usr/local/bin/gpio -g write 27 0"));
$filename = "antena.txt";
$handle = fopen($filename, "w");
fwrite($handle, "4");
fclose($handle);
echo '
<script type="text/javascript">
parent.navi.location.reload();
</script>';
?>
</body>
</html>
```

File: start.php

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title></title>
<script language="JavaScript">
function getDetails(obj)
{
    var clickX = window.event.x-obj.offsetLeft;
    var clickY = window.event.y-obj.offsetTop;
    clickX=200-clickX;
    clickY=clickY-200;
    var azi = clickY / clickX;
    azi = Math.atan(azi);
    azi=parseInt(90-57.29*azi);
    if (clickX>=0)
    {
        azi=180+azi;
    }
    document.getElementById("p_az").value = azi;
}

function draw()
{
    var canvas = document.getElementById('cav1');
    if (canvas.getContext)
    {
        var ctx = canvas.getContext('2d');
        ctx.drawImage(krug, 0, 0);
        ctx.beginPath();
        ctx.moveTo(200, 200);
        ctx.lineTo(azi_x, azi_y);
        ctx.strokeStyle = '#0000FF';
        ctx.lineWidth = 2;
        ctx.stroke();
        ctx.beginPath();
        ctx.moveTo(200, 200);
        ctx.lineTo(pos_x, pos_y);
        ctx.strokeStyle = '#FF00FF';
        ctx.lineWidth = 2;
        ctx.stroke();
    }
    else
```

```

{
    // make the element with the no-canvas message visible
    document.getElementById('no_cav').style.display = 'block';
}
}
</script>
</head>
<body onload="draw()">
<canvas width="400" height="400" id="cav1" onclick="getDetails(this)">
    
</canvas>
<p id="no_cav" style="display:none;">You need a Canvas-compatible browser to view canvas area.</p>
<?php
error_reporting(E_ALL);
ini_set('display_errors', '1');
require("php_serial.class.php");
$serial = new phpSerial;
$serial->deviceSet("/dev/ttyAMA0");
$serial->confBaudRate(9600);
$serial->confParity("none");
$serial->confCharacterLength(8);
$serial->confStopBits(1);
$serial->deviceOpen();
echo "<br>";
$filename = "azimut.txt";
$handle = fopen($filename, "r");
$azim_f = fread($handle, filesize($filename));
fclose($handle);
$azim_xy=(180-$azim_f)*0.01744;
$azim_x=120*(sin($azim_xy))+200;
$azim_y=120*(cos($azim_xy))+200;
$azim_x=(int)$azim_x;
$azim_y=(int)$azim_y;
echo '
<script type="text/javascript">
// <![CDATA[
    azi_x = " . $azim_x . ";
    azi_y = " . $azim_y . ";
// ]]>
</script>';
//print(" $azim_x $azim_y");
echo "<br>";
$serial->sendMessage("C\r\n");
$str = $serial->readPort();
$str = substr($str, 2, 3);
//$str = 150;
echo "Position: $str ($azim_f)";
echo "<br>";
echo '
<script type="text/javascript">
// <![CDATA[
    var pos = " . $str . ";
// ]]>
pos=(180-pos)*0.01744;
var pos_x=120*(Math.sin(pos))+200;
var pos_y=120*(Math.cos(pos))+200;
</script>';

if(!isset($_REQUEST['p_az']))
{
    ?>
    <form method="get" name="input">
        Azimuth: <input id="p_az" name="p_az" value="" style="width: 50px;" maxlength="3">
        <input id="GO" value="GO" type="submit">
    </form>
    <?php
    exit();
}
$azim = $_REQUEST['p_az'];
print("Azimuth = $azim");
if (strlen($azim)<1)

```

```

{
  $azim="0";
}
echo "<br>";
$filename = "azimut.txt";
$handle = fopen($filename, "w");
fwrite($handle, $azim);
fclose($handle);
$serial->sendMessage("M$azim\r\n");
$serial->deviceClose();
echo "<br>";
?>
<meta http-equiv="refresh" content="1;url=start.php">
</body>
</html>

```

File: readme.txt

```

Rottator Antenna Controller:
-----
C      - read azimuth
C2     - read azimuth elevation
Maaa   - write azimuth and go
S      - stop
Waaa   - write azimuth and go
Waaa eee - write azimuth elevation and go

```

File: antena.txt

```
1
```

File: azimuth.txt

```
231
```

File: php_serial.class.php

```

<?php
define ("SERIAL_DEVICE_NOTSET", 0);
define ("SERIAL_DEVICE_SET", 1);
define ("SERIAL_DEVICE_OPENED", 2);

/**
 * Serial port control class
 *
 * THIS PROGRAM COMES WITH ABSOLUTELY NO WARRANTIES !
 * USE IT AT YOUR OWN RISKS !
 *
 * @author Rémy Sanchez <thenux@gmail.com>
 * @thanks Aurélien Derouineau for finding how to open serial ports with windows
 * @thanks Alec Avedisyan for help and testing with reading
 * @copyright under GPL 2 licence
 */
class phpSerial
{
    var $_device = null;
    var $_windevice = null;
    var $_dHandle = null;
    var $_dState = SERIAL_DEVICE_NOTSET;
    var $_buffer = "";
    var $_os = "";

    /**
     * This var says if buffer should be flushed by sendMessage (true) or manually (false)
     *
     * @var bool
     */
    var $autoflush = true;

    /**
     * Constructor. Perform some checks about the OS and setserial
     *
     * @return phpSerial
     */
    function phpSerial ()
    {
        setlocale(LC_ALL, "en_US");

        $sysname = php_uname();

```



```

if (substr($sysname, 0, 5) === "Linux")
{
    $this->_os = "linux";

    if($this->_exec("stty --version") === 0)
    {
        register_shutdown_function(array($this, "deviceClose"));
    }
    else
    {
        trigger_error("No stty available, unable to run.", E_USER_ERROR);
    }
}
elseif(substr($sysname, 0, 7) === "Windows")
{
    $this->_os = "windows";
    register_shutdown_function(array($this, "deviceClose"));
}
else
{
    trigger_error("Host OS is neither linux nor windows, unable tu run.", E_USER_ERROR);
    exit();
}
}

//
// OPEN/CLOSE DEVICE SECTION -- {START}
//

/**
 * Device set function : used to set the device name/address.
 * -> linux : use the device address, like /dev/ttyS0
 * -> windows : use the COMxx device name, like COM1 (can also be used
 * with linux)
 *
 * @param string $device the name of the device to be used
 * @return bool
 */
function deviceSet ($device)
{
    if ($this->_dState !== SERIAL_DEVICE_OPENED)
    {
        if ($this->_os === "linux")
        {
            if (preg_match("@^AMA(\d+):?$@i", $device, $matches))
            {
                $device = "/dev/ttyAMA" . ($matches[1] - 1);
            }

            if ($this->_exec("stty -F " . $device) === 0)
            {
                $this->_device = $device;
                $this->_dState = SERIAL_DEVICE_SET;
                return true;
            }
        }
        elseif ($this->_os === "windows")
        {
            if (preg_match("@^COM(\d+):?$@i", $device, $matches) and $this->_exec(exec("mode " . $device)) === 0)
            {
                $this->_windevice = "COM" . $matches[1];
                $this->_device = "\\.\com" . $matches[1];
                $this->_dState = SERIAL_DEVICE_SET;
                return true;
            }
        }

        trigger_error("Specified serial port is not valid", E_USER_WARNING);
        return false;
    }
    else
    {
        trigger_error("You must close your device before to set an other one",
E_USER_WARNING);
        return false;
    }
}

/**

```

```

* Opens the device for reading and/or writing.
*
* @param string $mode Opening mode : same parameter as fopen()
* @return bool
*/
function deviceOpen ($mode = "r+b")
{
    if ($this->_dState === SERIAL_DEVICE_OPENED)
    {
        trigger_error("The device is already opened", E_USER_NOTICE);
        return true;
    }

    if ($this->_dState === SERIAL_DEVICE_NOTSET)
    {
        trigger_error("The device must be set before to be open", E_USER_WARNING);
        return false;
    }

    if (!preg_match("@^[raw]\+?b?$@", $mode))
    {
        trigger_error("Invalid opening mode : ".$mode.". Use fopen() modes.", E_USER_WARNING);
        return false;
    }

    $this->_dHandle = @fopen($this->_device, $mode);

    if ($this->_dHandle !== false)
    {
        stream_set_blocking($this->_dHandle, 0);
        $this->_dState = SERIAL_DEVICE_OPENED;
        return true;
    }

    $this->_dHandle = null;
    trigger_error("Unable to open the device", E_USER_WARNING);
    return false;
}

/**
* Closes the device
*
* @return bool
*/
function deviceClose ()
{
    if ($this->_dState !== SERIAL_DEVICE_OPENED)
    {
        return true;
    }

    if (fclose($this->_dHandle))
    {
        $this->_dHandle = null;
        $this->_dState = SERIAL_DEVICE_SET;
        return true;
    }

    trigger_error("Unable to close the device", E_USER_ERROR);
    return false;
}

//
// OPEN/CLOSE DEVICE SECTION -- {STOP}
//

//
// CONFIGURE SECTION -- {START}
//

/**
* Configure the Baud Rate
* Possible rates : 110, 150, 300, 600, 1200, 2400, 4800, 9600, 38400,
* 57600 and 115200.
*
* @param int $rate the rate to set the port in
* @return bool
*/
function confBaudRate ($rate)
{
    if ($this->_dState !== SERIAL_DEVICE_SET)

```

```

        {
            trigger_error("Unable to set the baud rate : the device is either not set or opened",
E_USER_WARNING);
            return false;
        }

        $validBauds = array (
            110    => 11,
            150    => 15,
            300    => 30,
            600    => 60,
            1200   => 12,
            2400   => 24,
            4800   => 48,
            9600   => 96,
            19200  => 19,
            38400  => 38400,
            57600  => 57600,
            115200 => 115200
        );

        if (isset($validBauds[$rate]))
        {
            if ($this->_os === "linux")
            {
                $ret = $this->_exec("stty -F " . $this->_device . " " . (int) $rate, $out);
            }
            elseif ($this->_os === "windows")
            {
                $ret = $this->_exec("mode " . $this->_windevice . " BAUD=" .
$validBauds[$rate], $out);
            }
            else return false;

            if ($ret !== 0)
            {
                trigger_error ("Unable to set baud rate: " . $out[1], E_USER_WARNING);
                return false;
            }
        }
    }

    /**
     * Configure parity.
     * Modes : odd, even, none
     *
     * @param string $parity one of the modes
     * @return bool
     */
    function confParity ($parity)
    {
        if ($this->_dState !== SERIAL_DEVICE_SET)
        {
            trigger_error("Unable to set parity : the device is either not set or opened",
E_USER_WARNING);
            return false;
        }

        $args = array(
            "none" => "-parenb",
            "odd"  => "parenb parodd",
            "even" => "parenb -parodd",
        );

        if (!isset($args[$parity]))
        {
            trigger_error("Parity mode not supported", E_USER_WARNING);
            return false;
        }

        if ($this->_os === "linux")
        {
            $ret = $this->_exec("stty -F " . $this->_device . " " . $args[$parity], $out);
        }
        else
        {
            $ret = $this->_exec("mode " . $this->_windevice . " PARITY=" . $parity{0}, $out);
        }

        if ($ret === 0)
        {

```

```

        return true;
    }

    trigger_error("Unable to set parity : " . $out[1], E_USER_WARNING);
    return false;
}

/**
 * Sets the length of a character.
 *
 * @param int $int length of a character (5 <= length <= 8)
 * @return bool
 */
function confCharacterLength ($int)
{
    if ($this->_dState !== SERIAL_DEVICE_SET)
    {
        trigger_error("Unable to set length of a character : the device is either not set or
opened", E_USER_WARNING);
        return false;
    }

    $int = (int) $int;
    if ($int < 5) $int = 5;
    elseif ($int > 8) $int = 8;

    if ($this->_os === "linux")
    {
        $ret = $this->_exec("stty -F " . $this->_device . " cs" . $int, $out);
    }
    else
    {
        $ret = $this->_exec("mode " . $this->_windevice . " DATA=" . $int, $out);
    }

    if ($ret === 0)
    {
        return true;
    }

    trigger_error("Unable to set character length : " . $out[1], E_USER_WARNING);
    return false;
}

/**
 * Sets the length of stop bits.
 *
 * @param float $length the length of a stop bit. It must be either 1,
 * 1.5 or 2. 1.5 is not supported under linux and on some computers.
 * @return bool
 */
function confStopBits ($length)
{
    if ($this->_dState !== SERIAL_DEVICE_SET)
    {
        trigger_error("Unable to set the length of a stop bit : the device is either not set
or opened", E_USER_WARNING);
        return false;
    }

    if ($length != 1 and $length != 2 and $length != 1.5 and !($length == 1.5 and $this->_os ===
"linux"))
    {
        trigger_error("Specified stop bit length is invalid", E_USER_WARNING);
        return false;
    }

    if ($this->_os === "linux")
    {
        $ret = $this->_exec("stty -F " . $this->_device . " " . (($length == 1) ? "-" : "") .
"cstopb", $out);
    }
    else
    {
        $ret = $this->_exec("mode " . $this->_windevice . " STOP=" . $length, $out);
    }

    if ($ret === 0)
    {
        return true;
    }
}

```

```

        trigger_error("Unable to set stop bit length : " . $out[1], E_USER_WARNING);
        return false;
    }

/**
 * Configures the flow control
 *
 * @param string $mode Set the flow control mode. Available modes :
 *     -> "none" : no flow control
 *     -> "rts/cts" : use RTS/CTS handshaking
 *     -> "xon/xoff" : use XON/XOFF protocol
 * @return bool
 */
function confFlowControl ($mode)
{
    if ($this->_dState !== SERIAL_DEVICE_SET)
    {
        trigger_error("Unable to set flow control mode : the device is either not set or
opened", E_USER_WARNING);
        return false;
    }

    $linuxModes = array(
        "none" => "clocal -crtcts -ixon -ixoff",
        "rts/cts" => "-clocal crtscts -ixon -ixoff",
        "xon/xoff" => "-clocal -crtcts ixon ixoff"
    );
    $windowsModes = array(
        "none" => "xon=off octs=off rts=on",
        "rts/cts" => "xon=off octs=on rts=hs",
        "xon/xoff" => "xon=on octs=off rts=on",
    );

    if ($mode !== "none" and $mode !== "rts/cts" and $mode !== "xon/xoff") {
        trigger_error("Invalid flow control mode specified", E_USER_ERROR);
        return false;
    }

    if ($this->_os === "linux")
        $ret = $this->_exec("stty -F " . $this->_device . " " . $linuxModes[$mode], $out);
    else
        $ret = $this->_exec("mode " . $this->_windevice . " " . $windowsModes[$mode], $out);

    if ($ret === 0) return true;
    else {
        trigger_error("Unable to set flow control : " . $out[1], E_USER_ERROR);
        return false;
    }
}

/**
 * Sets a setserial parameter (cf man setserial)
 * NO MORE USEFUL !
 *     -> No longer supported
 *     -> Only use it if you need it
 *
 * @param string $param parameter name
 * @param string $arg parameter value
 * @return bool
 */
function setSetserialFlag ($param, $arg = "")
{
    if (!$this->_ckOpened()) return false;

    $return = exec ("setserial " . $this->_device . " " . $param . " " . $arg . " 2>&1");

    if ($return{0} === "I")
    {
        trigger_error("setserial: Invalid flag", E_USER_WARNING);
        return false;
    }
    elseif ($return{0} === "/")
    {
        trigger_error("setserial: Error with device file", E_USER_WARNING);
        return false;
    }
    else
    {
        return true;
    }
}

```

```

}

//
// CONFIGURE SECTION -- {STOP}
//

//
// I/O SECTION -- {START}
//

/**
 * Sends a string to the device
 *
 * @param string $str string to be sent to the device
 * @param float $waitForReply time to wait for the reply (in seconds)
 */
function sendMessage ($str, $waitForReply = 0.1)
{
    $this->_buffer .= $str;

    if ($this->autoflush === true) $this->flush();

    usleep((int) ($waitForReply * 1000000));
}

/**
 * Reads the port until no new datas are available, then return the content.
 *
 * @param int $count number of characters to be read (will stop before
 * if less characters are in the buffer)
 * @return string
 */
function readPort ($count = 0)
{
    if ($this->_dState !== SERIAL_DEVICE_OPENED)
    {
        trigger_error("Device must be opened to read it", E_USER_WARNING);
        return false;
    }

    if ($this->_os === "linux")
    {
        $content = ""; $i = 0;

        if ($count !== 0)
        {
            do {
                if ($i > $count) $content .= fread($this->_dHandle, ($count - $i));
                else $content .= fread($this->_dHandle, 128);
            } while (($i += 128) === strlen($content));
        }
        else
        {
            do {
                $content .= fread($this->_dHandle, 128);
            } while (($i += 128) === strlen($content));
        }

        return $content;
    }
    elseif ($this->_os === "windows")
    {
        /* Do nohting : not implented yet */
    }

    trigger_error("Reading serial port is not implemented for Windows", E_USER_WARNING);
    return false;
}

/**
 * Flushes the output buffer
 *
 * @return bool
 */
function flush ()
{
    if (!$this->_ckOpened()) return false;

    if (fwrite($this->_dHandle, $this->_buffer) !== false)
    {
        $this->_buffer = "";
    }
}

```



```
        return true;
    }
    else
    {
        $this->_buffer = "";
        trigger_error("Error while sending message", E_USER_WARNING);
        return false;
    }
}

//
// I/O SECTION -- {STOP}
//

//
// INTERNAL TOOLKIT -- {START}
//

function _ckOpened()
{
    if ($this->_dState !== SERIAL_DEVICE_OPENED)
    {
        trigger_error("Device must be opened", E_USER_WARNING);
        return false;
    }

    return true;
}

function _ckClosed()
{
    if ($this->_dState !== SERIAL_DEVICE_CLOSED)
    {
        trigger_error("Device must be closed", E_USER_WARNING);
        return false;
    }

    return true;
}

function _exec($cmd, &$out = null)
{
    $desc = array(
        1 => array("pipe", "w"),
        2 => array("pipe", "w")
    );

    $proc = proc_open($cmd, $desc, $pipes);

    $ret = stream_get_contents($pipes[1]);
    $err = stream_get_contents($pipes[2]);

    fclose($pipes[1]);
    fclose($pipes[2]);

    $retVal = proc_close($proc);

    if (func_num_args() == 2) $out = array($ret, $err);
    return $retVal;
}

//
// INTERNAL TOOLKIT -- {STOP}
//
}
?>
```

File: uart_test.c

```

#include <stdio.h>
#include <unistd.h> //Used for UART
#include <fcntl.h> //Used for UART
#include <termios.h> //Used for UART

main()
{
    //-----
    //----- SETUP USART 0 -----
    //-----
    //At bootup, pins 8 and 10 are already set to UART0_TXD, UART0_RXD (ie the alt0 function)
    respectively
    int uart0_filestream = -1;

    //OPEN THE UART
    //The flags (defined in fcntl.h):
    //    Access modes (use 1 of these):
    //        O_RDONLY - Open for reading only.
    //        O_RDWR - Open for reading and writing.
    //        O_WRONLY - Open for writing only.
    //
    //    O_NDELAY / O_NONBLOCK (same function) - Enables nonblocking mode. When set read requests on
    the file can return immediately with a failure status
    //
    // immediately available (instead of blocking). Likewise, write requests can also return
    // immediately with a
    failure status if the output can't be written immediately.
    //
    //    O_NOCTTY - When set and path identifies a terminal device, open() shall not cause the terminal
    device to become the controlling terminal for the process.
    uart0_filestream = open("/dev/ttyAMA0", O_RDWR | O_NOCTTY | O_NDELAY); //Open in non
    blocking read/write mode
    if (uart0_filestream == -1)
    {
        //ERROR - CAN'T OPEN SERIAL PORT
        printf("Error - Unable to open UART. Ensure it is not in use by another application\n");
    }

    //CONFIGURE THE UART
    //The flags (defined in /usr/include/termios.h - see
    http://pubs.opengroup.org/onlinepubs/007908799/xsh/termios.h.html):
    //    Baud rate:- B1200, B2400, B4800, B9600, B19200, B38400, B57600, B115200,
    //    B230400, B460800, B500000, B576000, B921600, B1000000, B1152000, B1500000, B2000000, B2500000,
    B3000000, B3500000, B4000000
    //    CSIZE:- CS5, CS6, CS7, CS8
    //    CLOCAL - Ignore modem status lines
    //    CREAD - Enable receiver
    //    IGNPAR = Ignore characters with parity errors
    //    ICRNL - Map CR to NL on input (Use for ASCII comms where you want to auto correct end of line
    characters - don't use for binary comms!)
    //    PARENB - Parity enable
    //    PARODD - Odd parity (else even)
    struct termios options;
    tcgetattr(uart0_filestream, &options);
    options.c_cflag = B9600 | CS8 | CLOCAL | CREAD; //<Set baud rate
    options.c_iflag = IGNPAR;
    options.c_oflag = 0;
    options.c_lflag = 0;
    tcflush(uart0_filestream, TCIFLUSH);
    tcsetattr(uart0_filestream, TCSANOW, &options);

    //----- TX BYTES -----
    unsigned char tx_buffer[20];
    unsigned char *p_tx_buffer;

    p_tx_buffer = &tx_buffer[0];
    *p_tx_buffer++ = 'C';
    *p_tx_buffer++ = '\n';
    *p_tx_buffer++ = '\r';

    if (uart0_filestream != -1)
    {
        int count = write(uart0_filestream, &tx_buffer[0], (p_tx_buffer - &tx_buffer[0]));
    //Filestream, bytes to write, number of bytes to write
        if (count < 0)
        {
            printf("UART TX error\n");
        }
    }
}

```

```
//----- CHECK FOR ANY RX BYTES -----
if (uart0_filestream != -1)
{
    // Read up to 255 characters from the port if they are there
    unsigned char rx_buffer[256];
    int rx_length = read(uart0_filestream, (void*)rx_buffer, 255);           //Filestream, buffer
to store in, number of bytes to read (max)
    if (rx_length < 0)
    {
        //An error occured (will occur if there are no bytes)
    }
    else if (rx_length == 0)
    {
        //No data waiting
    }
    else
    {
        //Bytes received
        rx_buffer[rx_length] = '\0';
        printf("%i bytes read : %s\n", rx_length, rx_buffer);
    }
}

//----- CLOSE THE UART -----
close(uart0_filestream);
}
```

REFERENCE:

<http://www.digistore.rs/>
<http://www.raspberrypi.org/>
<http://www.raspberrypi-projects.com/pi/programming-in-c/uart-serial-port/using-the-uart>
<http://wiringpi.com/>
http://elinux.org/RPi_Serial_Connection
<http://www.hobbytronics.co.uk/raspberry-pi-serial-port>
<http://www.raspberrypi-spy.co.uk/2013/12/free-your-raspberry-pi-serial-port/>
<https://www.microchip.com/>
<http://www.g6lvb.com/articles/lvtracker/>
<http://www.foxdelta.com/products/st2-rs232.htm>