ARDUINO Antenna Rotator

Henry Wyatt VK4BUD

Aim: To design and build an antenna rotator for a small antenna.

Cost: Was minimal, since an Arduino (for the controller part) was already constructed (from another project - SLA battery tester) that had an LCD display (Appendix 2) and two robust relays connected, as well as an LM358 op amp (that was previously used to condition readings of analog voltage and current of the batteries tested).

Additional Parts for Controller:

Add a 2 pole momentary switch for Clockwise and Anticlockwise direction control.

Add a 5V DC regulator for the supply to the pot located at the rotator.

Add a pot at the rotator, analog values representing antenna direction fed back to the Arduino. A 5kΩ (3 turn) pot was already in the junk box. The pot needed to have 360 degree rotation as the gear on the main shaft and the gear on the pot would have the same number of teeth. The middle third of the pot would be used and would not get damaged if swung past the end points (0 to 360 degrees).

Parts Not Required:

Rotators normally have a brake, however the proposed motor had 3000:1 ratio gearbox, so the antenna could not turn the motor, so no brake required.

Because the Arduino is continuously reading the pot voltage, no limit switches were required at the extremes of travel, as the Arduino would remove power from the motor at the extremes.

Parts Required for Rotator:

Everything new was needed for the mechanicals at the Rotator, listed below:

Part	**Model Number**	**Source**
DC Motor (see Appendix 5)	417-9649	RS Components
Metalmate 30x30x1.6 x1m galv steel square tube		Bunnings
Metalmate 25x1mm x1m Aluminium round tube		Bunnings
6.3mm x1m Aluminium rod		Bunnings
Richmond Precision Ball Bearing 25mm x2		Bunnings
Stainless steel Hose Clamp x2 65-89mm		Bunnings
P&N Carbon Tap & Drill Set M4		Bunnings
Solid Brass Coupler x2	YG2600	Jaycar
Aluminium Hub x2	YG2784	Jaycar
Switch TGL mini ctr-off DPDT mom	PP0512	Jaycar
6mmx50mm D-Shaft	2101-0006-0050	Core Electronics
6mmx30mm D-Shaft	2121-0006-0030	Core Electronics

Flexible Clamping Shaft Coupler	4002-0006-0250	Core Electronics
Brass Pinion Gear 6mm D-Bore	2301-0006-0024	Core Electronics
Brass Pinion Gear 6mm Bore	2304-0006-0024	Core Electronics

Controller

The circuit of the controller is in Appendix 1.

Programming of the Arduino was the easy part with the pot attached directly to the box screw terminals for testing (see Appendix 3). Programming was via the Arduino Integrated Development Environment (IDE). The controller is stand-alone with Azimuth displayed on the LCD.

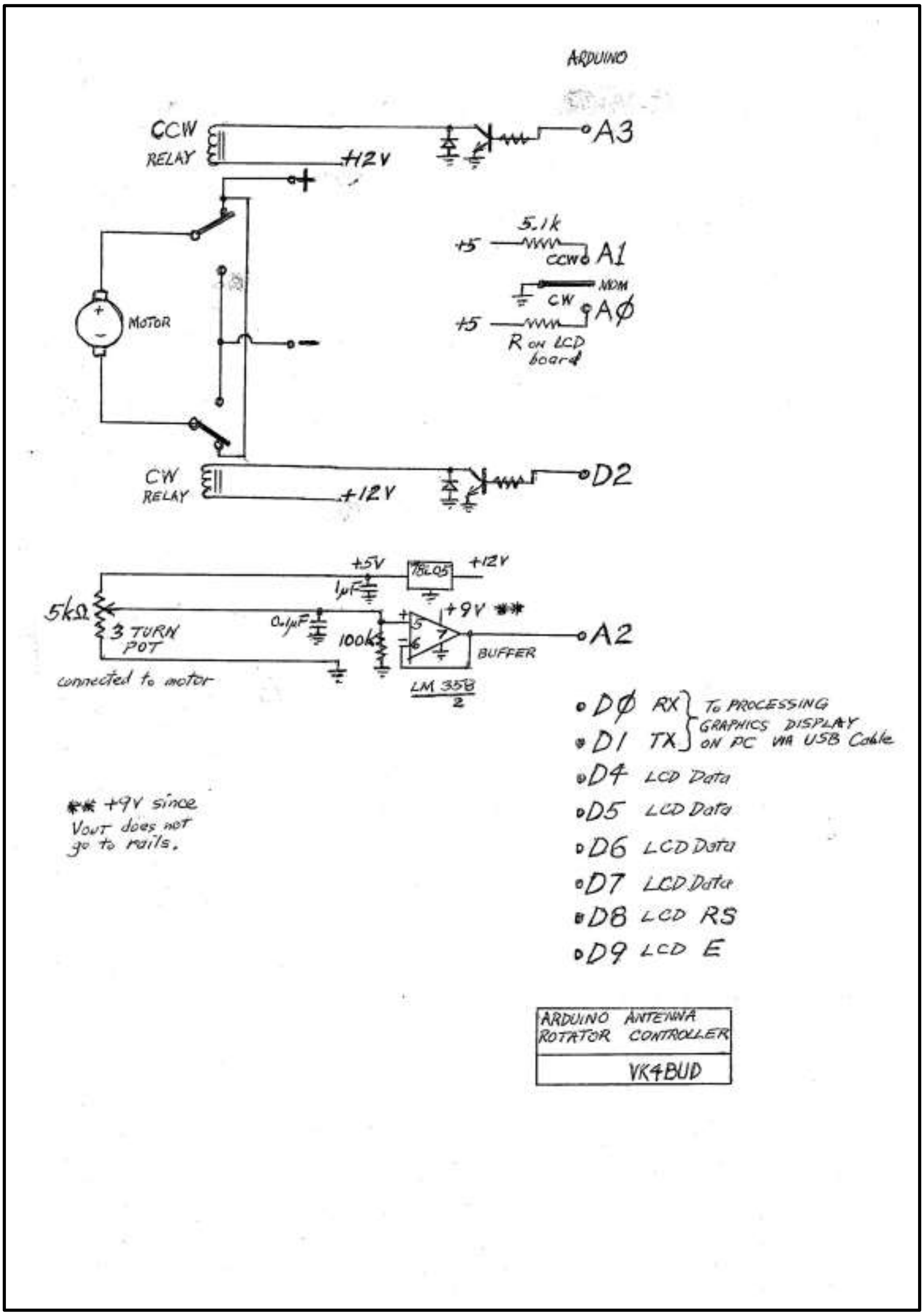
Rotator

1. A length of round aluminium tube is the support for the antenna. Force fit onto this are the two ball bearings. Notches for the bearings are cut into the steel square tube. The bearings are then secured with the hose clamps.
2. The aluminium hubs (which fit inside the round tube) are then tapped with a 4mm thread.
   Holes are drilled in the round tube and the hubs are screwed in with 4mm screws.
   A flexible shaft coupler handles any asymmetrical motion.
3. The motor is mounted on a steel plate, with a section of the plate bent to support the pot at a precise distance from the motor shaft.
4. The gears are assembled on the intermediate shafts.
5. Plumbing pipe will be used to weather-proof the motor assembly.

Display on a separate PC Screen

A compass program was adapted from the Internet. This uses the Processing IDE. The program runs on the PC and communicates with the Arduino via the USB cable (see Appendix 4). The Graphics png files were as per original, and one file had my callsign added roughly using Paint.

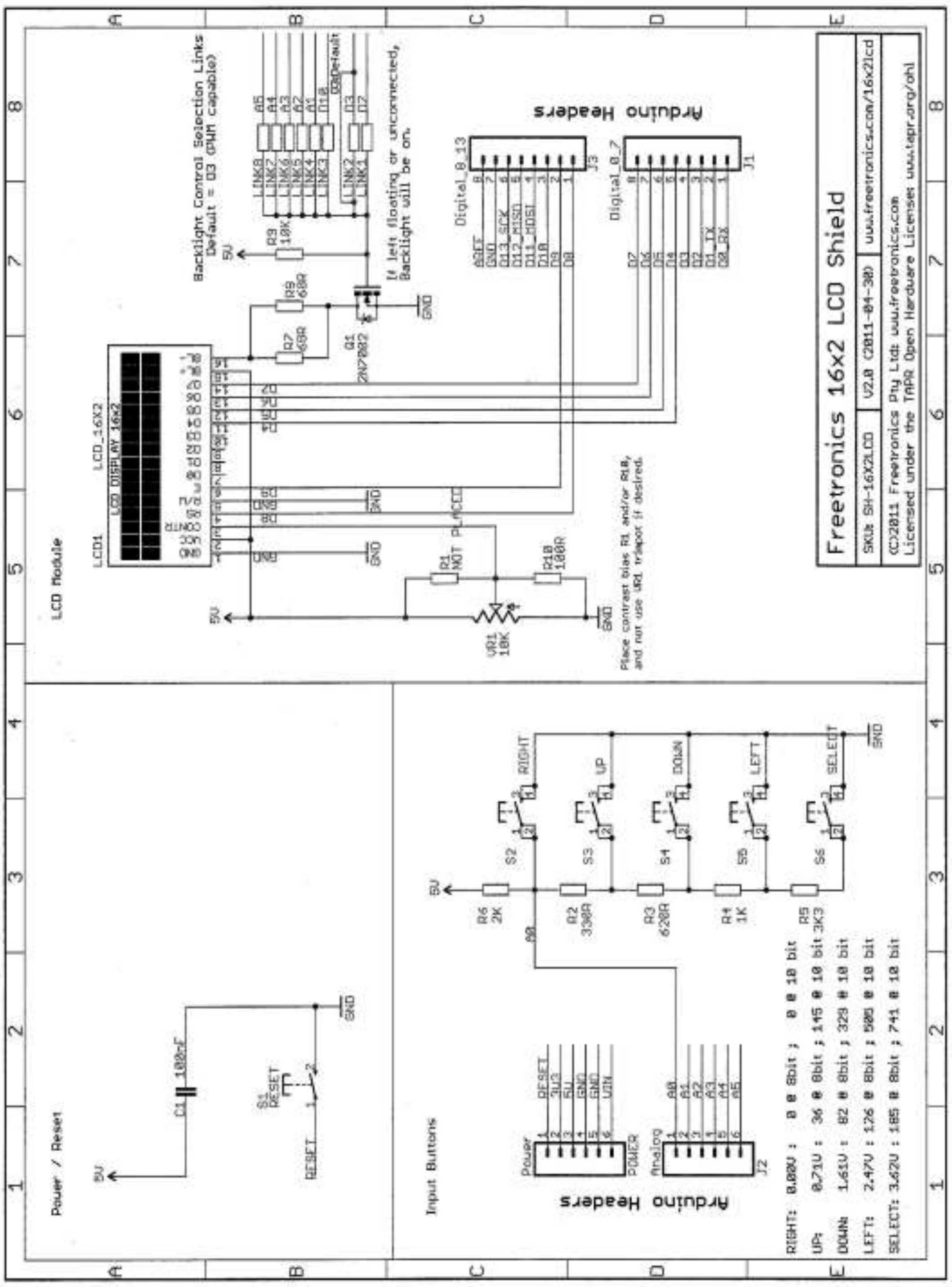
Appendix 1



Controller wiring. One plug pack fed a 12V regulator, a 9V regulator, and a 5V regulator.

A separate plug pack with selectable DC voltages fed the motor to avoid supply surges on the controller.

Appendix 2


Freetronics 16x2 LCD Shield
SKU: SH-16X2LCD
V2.0 (2011-04-30)
www.freetronics.com/16x2lcd
(C)2011 Freetronics Pty Ltd: www.freetronics.com
Licensed under the TAPR Open Hardware License: www.tapr.org/ohl
LCD Module
LCD1
LCD_16X2
LCD DISPLAY 16x2
Backlight Control Selection Links
Default = D3 (PWM capable)
If left floating or unconnected, Backlight will be on.
Q1
2N7002
R7
68R
R8
68R
R9
10K
R1
NOT PLACED
R10
100R
VR1
10K
Place contrast bias R1 and/or R10, and not use VR1 trimpot if desired.
Arduino Headers
Digital_0_7
Digital_8_13
J1
J3
Power / Reset
C1
100nF
S1
RESET
Input Buttons
RIGHT
UP
DOWN
LEFT
SELECT
R6
2K
R2
330R
R3
620R
R4
1K
R5
3K3
POWER
Analog
J2
RIGHT: 0.00V : 0 @ 8bit ; 0 @ 10 bit
UP: 0.71V : 36 @ 8bit ; 145 @ 10 bit
DOWN: 1.61V : 82 @ 8bit ; 329 @ 10 bit
LEFT: 2.47V : 126 @ 8bit ; 505 @ 10 bit
SELECT: 3.62V : 185 @ 8bit ; 741 @ 10 bit

LCD Module, plugs into a standard Arduino UNO board.

Appendix 3

```
/*   Arduino Compass VK4BUD
 *
   Antenna Rotator Controller Program
   written by Henry Wyatt July 2023
 *
   A2: Voltage back from Rotator pot (3 turn 5kΩ)
   D2: CW Relay
   A3: CCW Relay
   ADC voltages for the 5 buttons on analog input pin A0:
   RIGHT:  0.00V :     0 @ 10 bit
   UP:     0.71V :   145 @ 10 bit
   DOWN:   1.61V :   329 @ 10 bit
   LEFT:   2.47V :   505 @ 10 bit
   SELECT: 3.62V :   741 @ 10 bit
 *
*/
   //    DECLARATIONS
   #include <LiquidCrystal.h>
   LiquidCrystal lcd( 8, 9, 4, 5, 6, 7 );
   unsigned long startTime;
   unsigned long elapsedTime;
   unsigned int buttonVoltage;
   float headingDegrees, headingFiltered;
   int voltage;
   float actualvoltage;
   int voltage1;
   boolean CW;
   boolean CCW;

// ADC readings expected for the 5 buttons on the A0 ADC input
#define RIGHT_10BIT_ADC          0  // right
```

```
#define UP_10BIT_ADC          145  // up
#define DOWN_10BIT_ADC         329  // down
#define LEFT_10BIT_ADC        505  // left
#define SELECT_10BIT_ADC       741  // RESET button
#define BUTTONHYSTERESIS       10   // hysteresis for valid button sensing window
//return values for ReadButtons()
#define BUTTON_NONE            0  //
#define BUTTON_RIGHT           1  //
#define BUTTON_UP              2  //
#define BUTTON_DOWN            3  //
#define BUTTON_LEFT            4  //
#define BUTTON_SELECT          5  // RESET
byte button = BUTTON_NONE;      // return no button pressed if the below checks don't write to button


  //    SET UP
  void setup()
  {
   pinMode(A0, INPUT);           //CW Control
   pinMode(A1, INPUT);           //CCW Control
   pinMode(A2, INPUT);           //Voltage back from Rotator pot (3 turn 5kΩ)
   pinMode( 2, OUTPUT );          //CW Relay
   pinMode(A3, OUTPUT);           //CCW Relay
   delay(1000);
   digitalWrite(2, LOW);         //ensure CW Relay inactive
   digitalWrite( A3, LOW);        //ensure CCW Relay inactive

   lcd.begin(16,2);


   Serial.begin(9600);
   while (!Serial) {
   ; // wait for serial port to connect.
                }
  }
```

```
//   MAIN PROGRAM

void loop()

{

  lcd.setCursor(0,0);

  lcd.print("Antenna  Azimuth");

  lcd.setCursor(9,1);

  lcd.print("Degrees");

  delay(10);


voltage = analogRead(A2);

CW = analogRead(A0);

CCW = analogRead(A1);

delay (10);


//-------------------------------------------------------------------------

  if ((CW == LOW) && (voltage <= 673))  // CW

  {

    do {

   digitalWrite(2, HIGH);             // YELLOW LED CW        CW   POWER to Motor

   delay (10);

   CW = analogRead(A0);               // Read again check still pressed


   // Get real voltage

voltage = analogRead(A2);                  // actual digital value, 5V to pot

//voltage1 = map(voltage,337,673,0,1023);       // because 3 turn pot, only using 1 turn of 3 turn pot

voltage1 = map(voltage,337,673,1023,0);

delay(2);

//actualvoltage = voltage1*(5.0/1023.0);       // voltage 0 to 5V, mapped to middle 1 turn of pot

//actualvoltage = (actualvoltage-1.65)*3.0;    // old calc previous


//  PRINTOUT VALUES TO PROCESSING
```

```
  delay (10);  //  Wait 10 mseconds for next reading
    //Calculating Heading

// headingDegrees = voltage1*(360.0/1023.0); // The heading in Degrees units
 headingDegrees = ((voltage1/1023.0)*360.0);
 // Smoothing the output angle / Low pass filter
 headingFiltered = headingFiltered*0.60 + headingDegrees*0.40;
 lcd.setCursor(0,1);
 lcd.print("       ");
 lcd.setCursor(1,1);
 lcd.print(headingFiltered);

 //Sending the heading value through the Serial Port to Processing IDE
 Serial.println(headingFiltered);

 delay(50);
       }while ((CW == LOW)  && (voltage <= 673));  //End of CW control
   }

    digitalWrite(2, LOW);    //No more power to motor in CW direction
```

CW   Motor Stop

```
//------------------------------------------------------------------------



   if ((CCW == LOW) && (voltage >= 337))  // CCW
    {
     do {
     digitalWrite(A3, HIGH);            // RED LED CCW
```

CCW   POWER to Motor

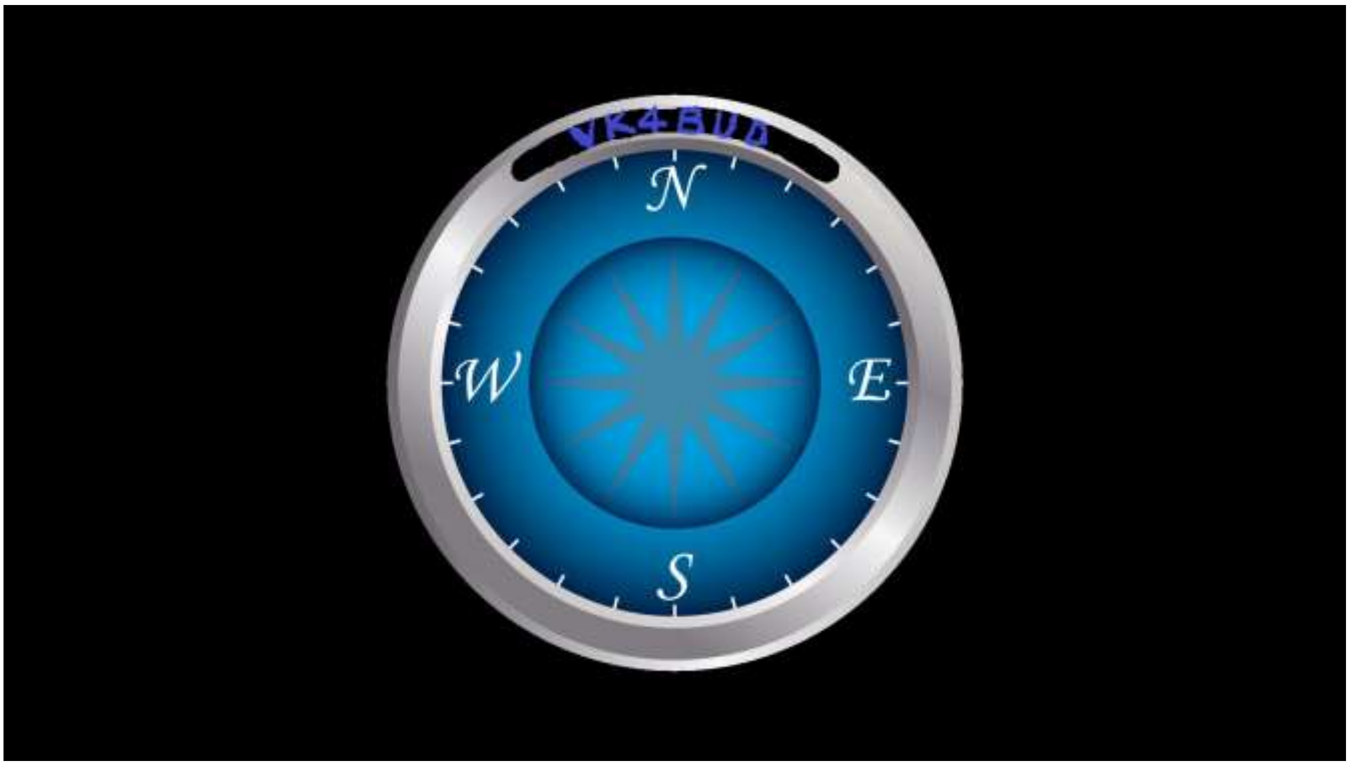
```
     delay (10);
     CCW = analogRead(A1);              // Read again check still pressed

     // Get real voltage
```

```
  voltage = analogRead(A2);                  // actual digital value, 5V to pot

  //voltage1 = map(voltage,337,673,0,1023);       // because 3 turn pot, only using 1 turn of 3 turn pot

  voltage1 = map(voltage,337,673,1023,0);

  delay(2);

  //actualvoltage = voltage1*(5.0/1023.0);       // voltage 0 to 5V, mapped to middle 1 turn of pot

  //actualvoltage = (actualvoltage-1.65)*3.0;    // old calc previous


  //  PRINTOUT VALUES TO PROCESSING


  delay (10);  //  Wait 10 mseconds for next reading

    //Calculating Heading


// headingDegrees = voltage1*(360.0/1023.0); // The heading in Degrees units

 headingDegrees = ((voltage1/1023.0)*360.0);

 // Smoothing the output angle / Low pass filter

 headingFiltered = headingFiltered*0.60 + headingDegrees*0.40;

 lcd.setCursor(0,1);

 lcd.print("       ");

 lcd.setCursor(1,1);

 lcd.print(headingFiltered);


 //Sending the heading value through the Serial Port to Processing IDE

 Serial.println(headingFiltered);


 delay(50);

       }while ((CCW == LOW)  && (voltage >= 337));  //End of CCW control

   }


   digitalWrite(A3, LOW);    //No more power to motor in CCW direction   CCW   Motor Stop

//-----------------------------------------------------------------------


 // Get real voltage
```

```
  delay(50);
  voltage = analogRead(A2);                    // actual digital value, 5V to pot
  //voltage1 = map(voltage,337,673,0,1023);       // because 3 turn pot, only using 1 turn of 3 turn pot
  voltage1 = map(voltage,337,673,1023,0);
  delay(2);
  //actualvoltage = voltage1*(5.0/1023.0);        // voltage 0 to 5V, mapped to middle 1 turn of pot
  //actualvoltage = (actualvoltage-1.65)*3.0;    // old calc previous

  //  PRINTOUT VALUES TO PROCESSING


  delay (10);  //  Wait 10 mseconds for next reading
    //Calculating Heading

// headingDegrees = voltage1*(360.0/1023.0); // The heading in Degrees units
 headingDegrees = ((voltage1/1023.0)*360.0);
 // Smoothing the output angle / Low pass filter
 headingFiltered = headingFiltered*0.60 + headingDegrees*0.40;
 lcd.setCursor(0,1);
 lcd.print("       ");
 lcd.setCursor(1,1);
 lcd.print(headingFiltered);

 //Sending the heading value through the Serial Port to Processing IDE
 Serial.println(headingFiltered);
 delay(50);



 }
```

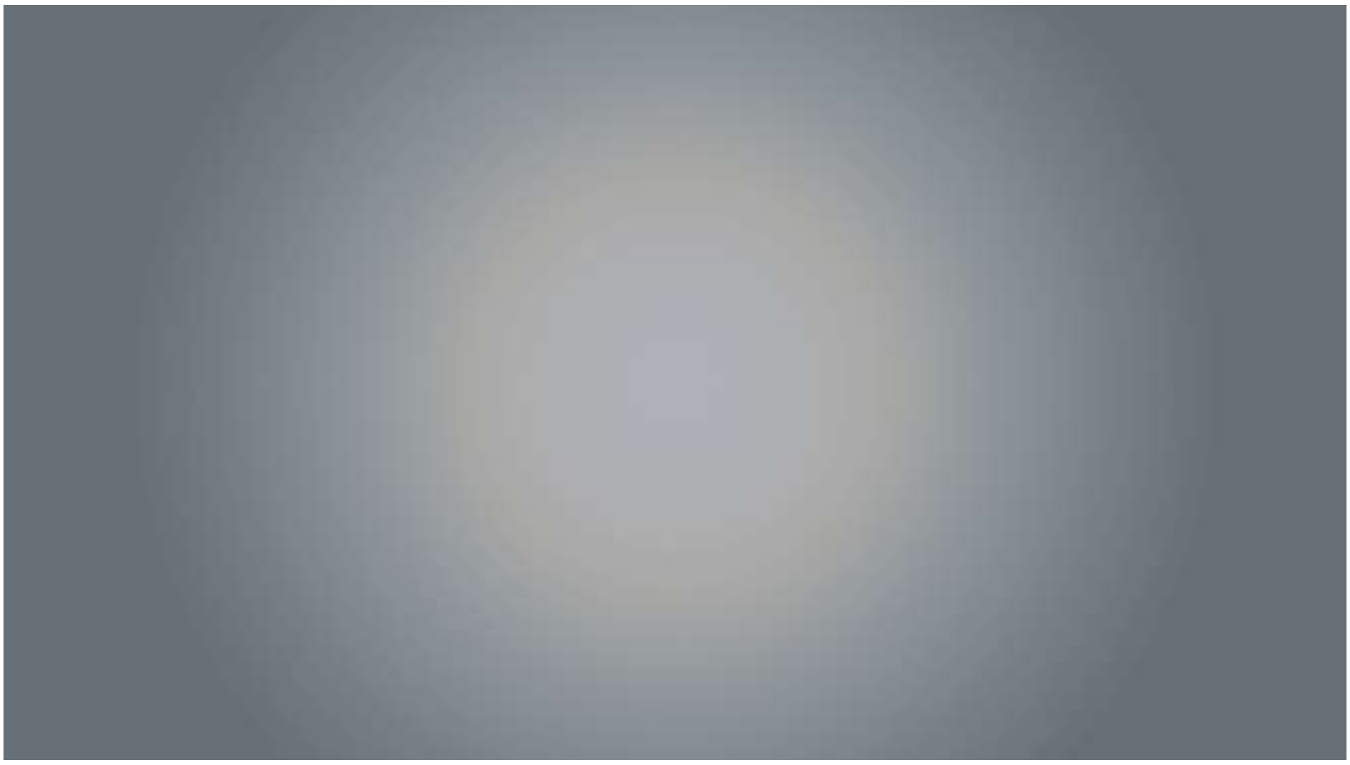
Appendix 4 **(with acknowledgement to original author)**

```
/*   Arduino Compass
 *
 *  by Dejan Nedelkovski,
 *  www.HowToMechatronics.com
 *
 */

import processing.serial.*;
import java.awt.event.KeyEvent;
import java.io.IOException;

Serial myPort;
PImage imgCompass;
PImage imgCompassArrow;
PImage background;

String data="";
float heading;

void setup() {
  size (1920, 1080, P3D);
  smooth();
  imgCompass = loadImage("Compass2.png");
  imgCompassArrow = loadImage("CompassArrow.png");
  background = loadImage("Background.png");

  myPort = new Serial(this, "COM8", 9600); // starts the serial communication
  myPort.bufferUntil('\n');
}

void draw() {

  image(background,0, 0); // Loads the Background image

  pushMatrix();
  translate(width/2, height/2, 0); // Translates the coordinate system into the center of the screen, so that the
rotation happen right in the center
  rotateZ(radians(-heading)); // Rotates the Compass around Z - Axis
  image(imgCompass, -960, -540); // Loads the Compass image and as the coordinate system is relocated we
need need to set the image at -960x, -540y (half the screen size)
  popMatrix(); // Brings coordinate system is back to the original position 0,0,0

  image(imgCompassArrow,0, 0); // Loads the CompassArrow image which is not affected by the rotateZ()
function because of the popMatrix() function
  textSize(100);
  text("Heading: " + heading,40,140); // Prints the value of the heading on the screen

  delay(40);

}

// starts reading data from the Serial Port
 void serialEvent (Serial myPort) {

   data = myPort.readStringUntil('\n');// reads the data from the Serial Port and puts it into the String variable
"data".

  //heading = float(data); // Converting the the String value into Float value
  heading = int(data);
}
```



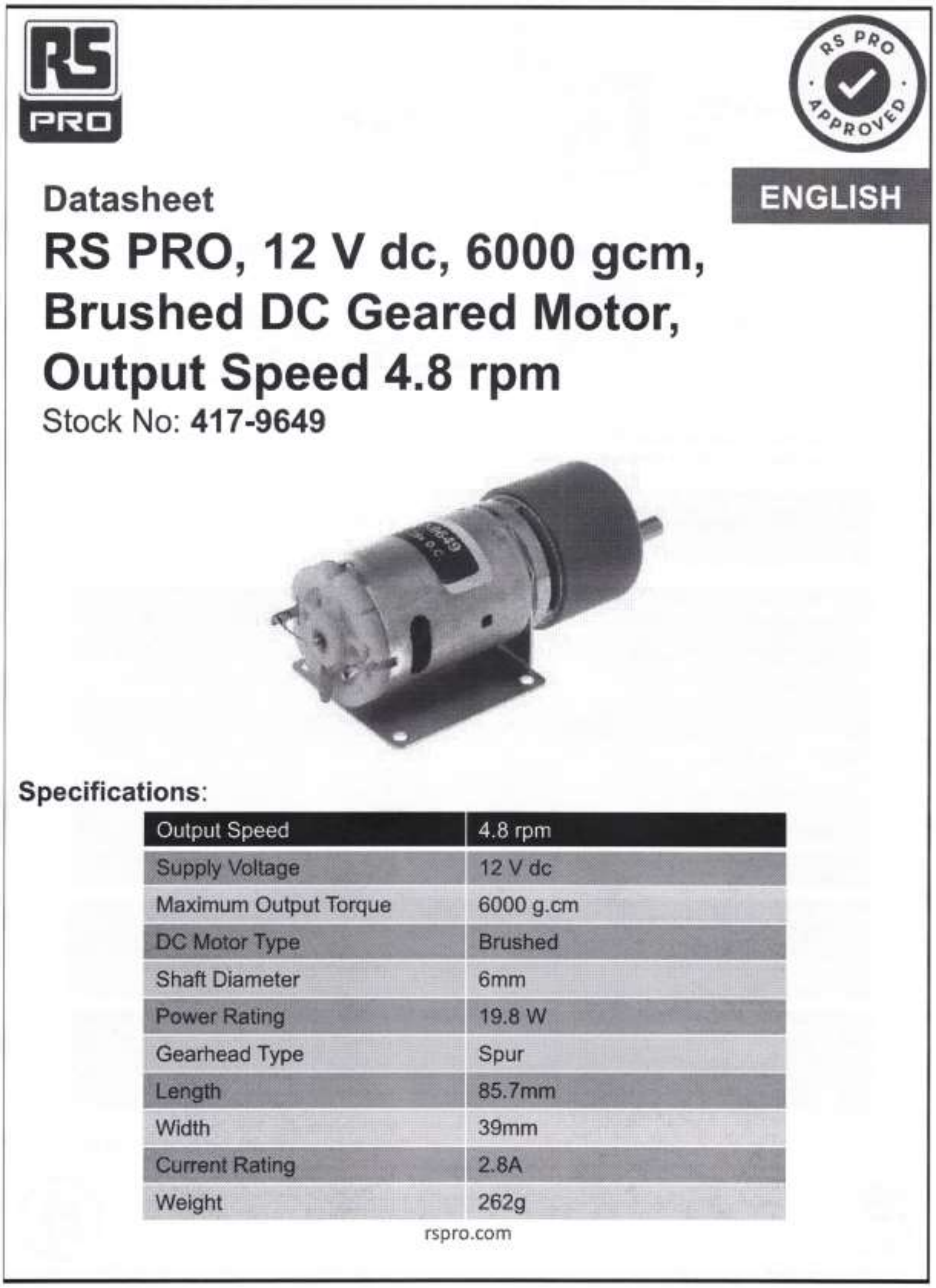
Compass2.png

CompassArrow.png



Background.png

Appendix 5



ENGLISH

Datasheet

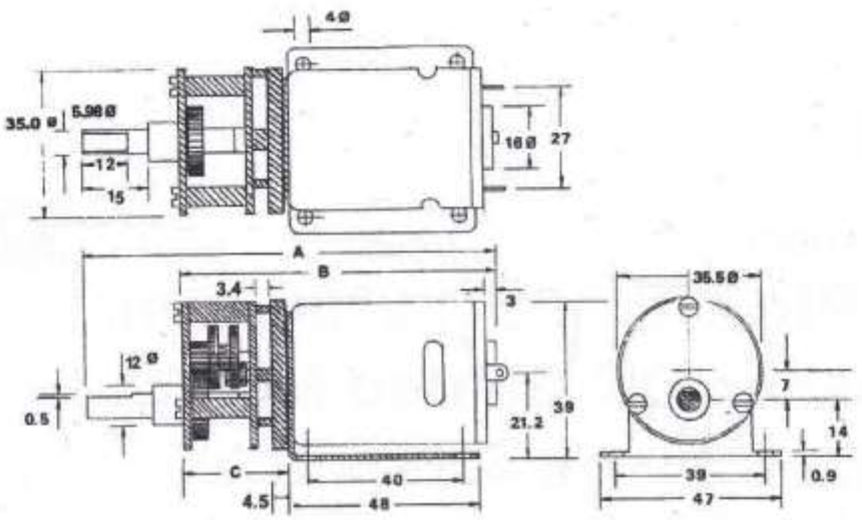
RS PRO, 12 V dc, 6000 gcm, Brushed DC Geared Motor, Output Speed 4.8 rpm

Stock No: **417-9649**

Specifications:

Output Speed	4.8 rpm
Supply Voltage	12 V dc
Maximum Output Torque	6000 g.cm
DC Motor Type	Brushed
Shaft Diameter	6mm
Power Rating	19.8 W
Gearhead Type	Spur
Length	85.7mm
Width	39mm
Current Rating	2.8A
Weight	262g

rspro.com



Operating relative humidity 20% ~ 85%
Operating temperature range -10°C ~ +60°C

Ratio	A	B	C
3000:1	110.7mm	85.7mm	33.7mm

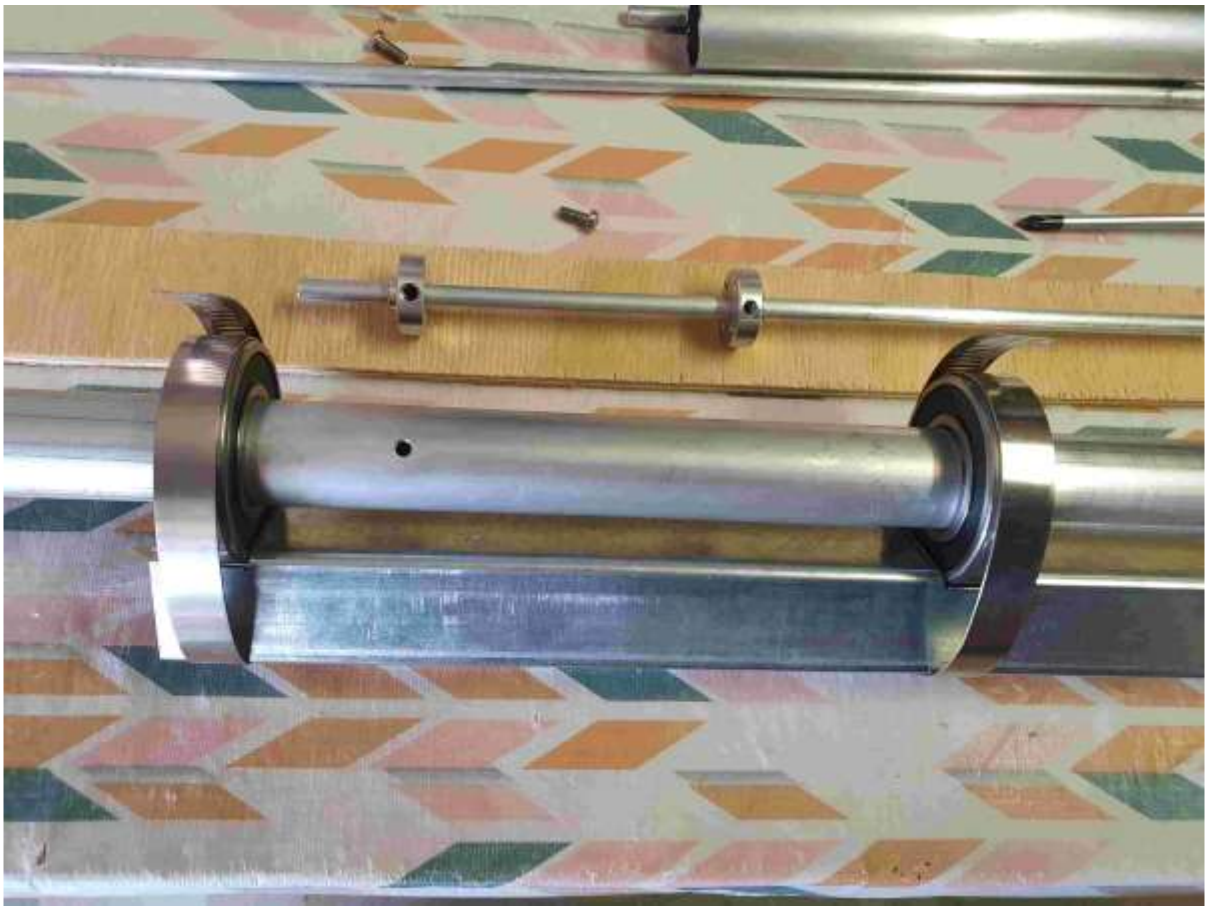
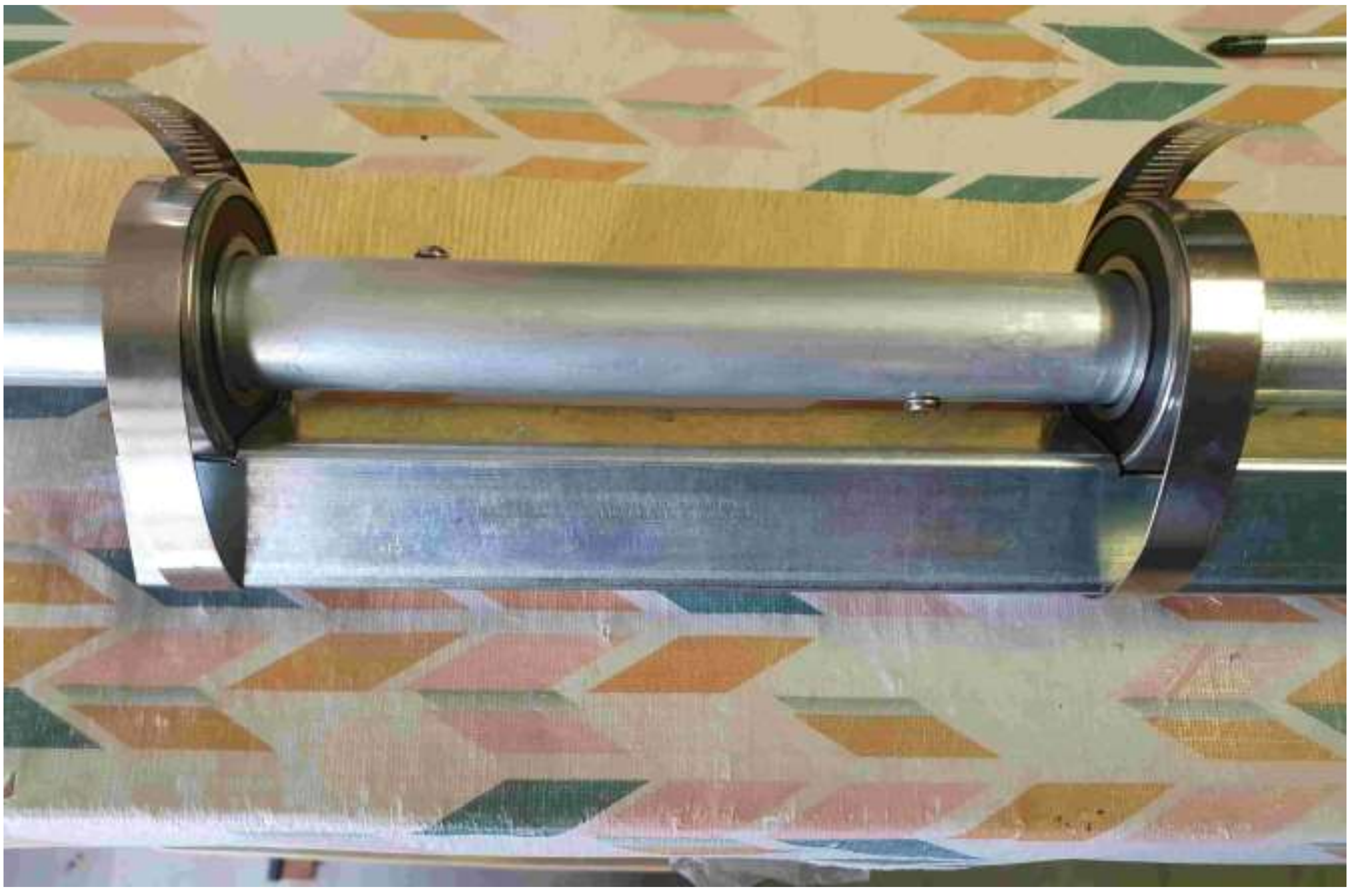
Gearbox Housing material	Metal
Backlash at no-load	< 2°
Bearing at output	Sleeve bearings
Radial load (10mm from flange)	< 1 kgf
Shaft axial load	< 0.7 kgf
Shaft press fit force max	< 20 kgf
Radial play of shaft	< 0.05mm
Thrust play of shaft	< 0.35mm

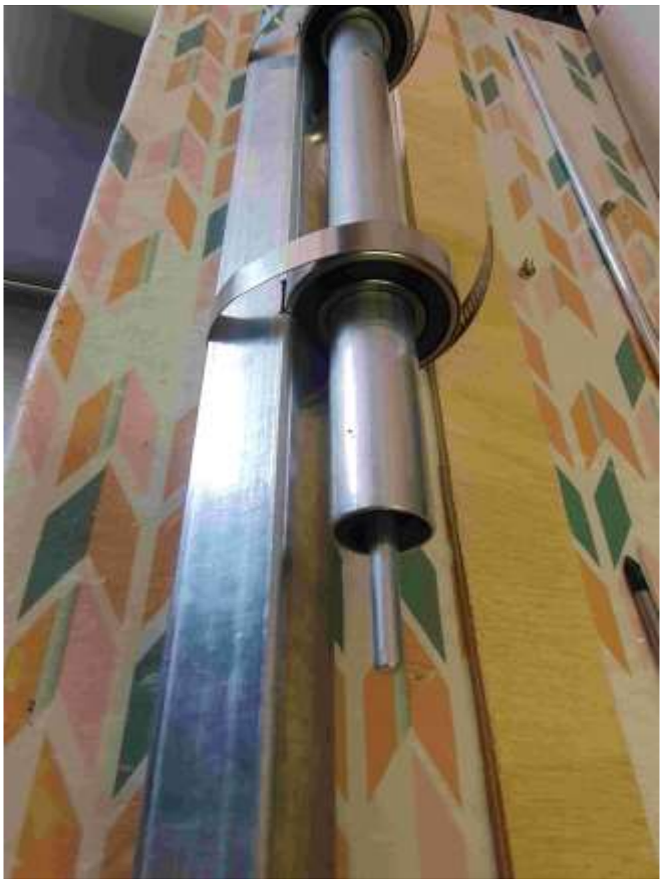
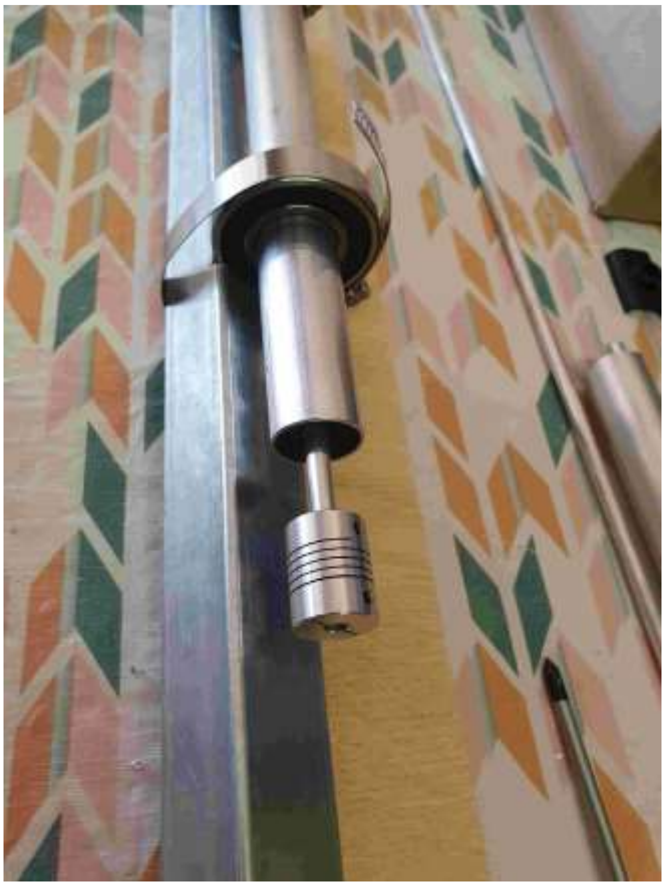
Reduction Ratio	Rated tolerance Torque	Max momentary Tolerance Torque	Efficiency
1/3000	6 kgf-cm Max.	18 kgf-cm	48%

Reduction Table RPM SUPPLY VOLTAGE	4.5v	6v	9v	12v	15v
417-9649	1.8	2.4	3.6	4.8	6

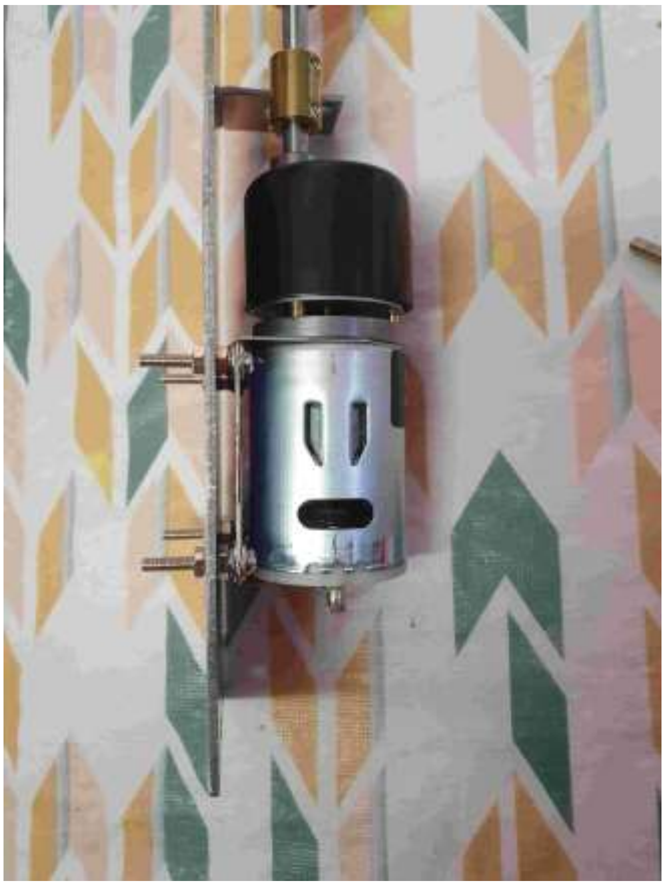
Note: Motor speeds may vary by (+) or (-) 12.5%

Appendix 6

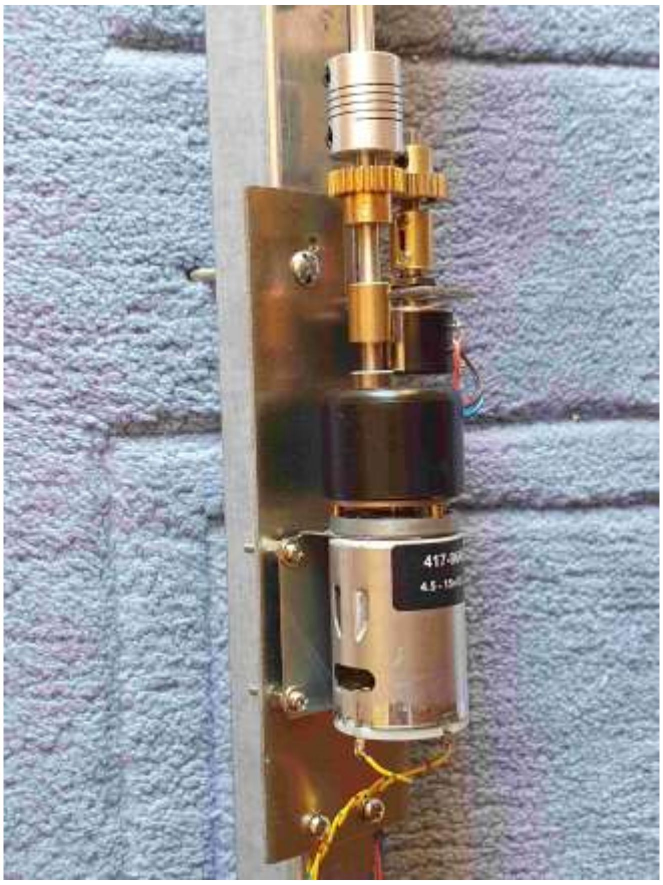



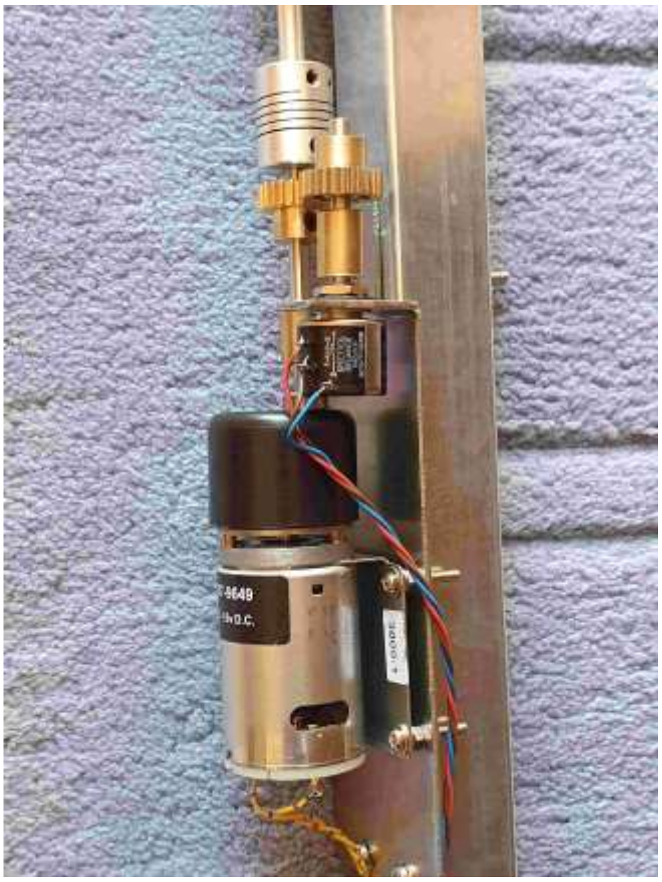



Flexible Shaft Coupler



WORKSHOP
CARBON TAP
& DRILL SET
M4
X 0.7
TAPER LEAD
P&N


417-66
4.5 - 15v


-6649
5v D.C.