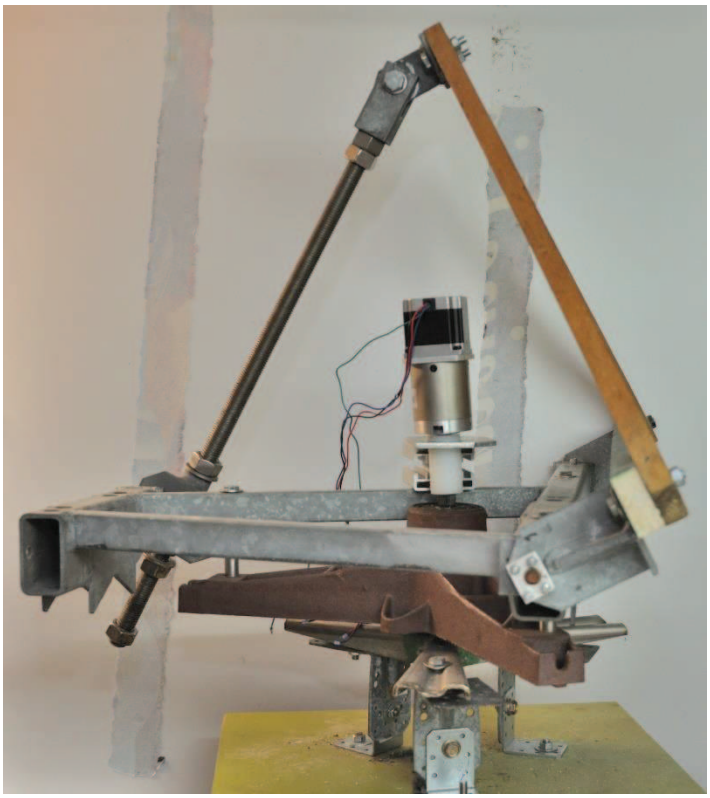
Präzise und reproduzierbare Ansteuerung mit Schrittmotoren



Kristel Boeijink (17)
Evelyn Gebel (17)
Malin Reuter (16)

Wettbewerb „Jugend Forscht“ 2017
Arbeitsgemeinschaft „Jugend Forscht“ des Christian Gymnasium Hermannsburg
Betreuung: StD Thomas Biedermann

Inhaltsverzeichnis

1. Einleitung

1.1. Intention

Seit über drei Jahren existiert am Christian-Gymnasium in Hermannsburg die Radioastronomie-AG, in der wir drei neben der Jugend Forscht-AG aktiv tätig sind. Es ist der AG bereits gelungen ein voll funktionsfähiges Radioteleskop mit einem Parabolspiegeldurchmesser von 2,5 m zu bauen und damit radioastronomische Messungen durchzuführen.
Durch eine enge Zusammenarbeit mit dem Max-Planck-Institut für Radioastronomie entstand daraufhin unser aktuelles AG- Projekt, ein Radioteleskop mit einem Parabolspiegeldurchmesser von 4 m. Außerdem nehmen wir im Rahmen der Jugend Forscht-AG als Aussteller an der Ideen Expo 2017 teil, sodass wir zusätzlich zu Anschauungszwecken ein Radioteleskop mit einem Parabolspiegel des Durchmessers 1,8 m betriebsfähig machen wollen.
Die beiden kleineren Reflektoren verfügen bislang über Antriebssysteme die auf Gleichstrommotoren basieren. Diese haben den Nachteil, dass ihre Ansteuerung und Positionierung mit einem hohen elektronischen Aufwand verbunden sind. Außerdem müssen sie mit einer bestimmten Mindestgeschwindigkeit gefahren werden. Aus diesem Grund ist geplant, den 4 m Reflektor mit leistungsstarken Schrittmotoren mit nach geschaltetem Planetengetriebe zu betreiben. Diese haben den Vorteil, dass sie eine viel höhere Positionsgenauigkeit erreichen und beliebig langsam gefahren werden können, was zum Beispiel für die Nachführung von Himmelsobjekten besonders wichtig ist. Um diese neue Antriebsart zu erproben, wird in unserem Projekt zunächst der 1,8 m Reflektor auf Schrittmotorantriebe umgerüstet. Die von uns dabei verwendete Motorsteuerung mit einem STM Microcontroller kann gleichzeitig verwendet werden, um einen auf Schrittmotorantrieb umgerüsteten Linearantrieb für den 4 m Reflektor zu testen.

1.2. Relevanz einer präzisen Steuerung

Radioteleskope nutzen Dipolantennen, um aus dem Universum elektromagnetische Strahlungen empfangen zu können. Da diese jedoch im Vergleich zu terrestrischer Strahlung äußerst schwach sind, bedarf es einer reproduzierbaren und sehr genauen Ansteuerung der Strahlungsquelle. Deshalb versuchen wir in unserem Projekt eine effiziente Steuerung der Teleskope durch selbstkonstruierte Antriebssysteme zu schaffen, deren Präzision wir mit einem Testprogramm, welches später als Steuerprogramm eingesetzt werden soll, überprüfen.

2. Mechanik

2.1. Schrittmotoren-Vorteile und Funktionsweise (Abb. 1)

Ein Schrittmotor hat zwei entscheidende Bestandteile. Zum einen ein Stabmagnet, um den kreisförmig vier Spulen angeordnet sind. Legt man an diese eine Spannung an, wird so ein Magnetfeld erzeugt. Nur eine der Spulen anzuschließen, sorgt dafür, dass der Magnet angezogen wird und sich folglich in eine bestimmte Position begibt. Schaltet man eine zweite danebenliegende Spule ein, bewegt er sich zwischen die beiden Spulen. Wird nun die erste Spule wieder abgestellt, bewegt sich der Stabmagnet noch ein Stück weiter, sodass er direkt auf die zweite Spule zeigt.
Wenn man dieses System fortsetzt, kann man den Stabmagneten und damit die Achse des Motors, in eine Drehbewegung versetzen. Bei dieser Bewegung bleibt genau bekannt, wie viele Schritte der Motor gegangen ist, wodurch wir eine genaue Positionsrückmeldung erhalten können.

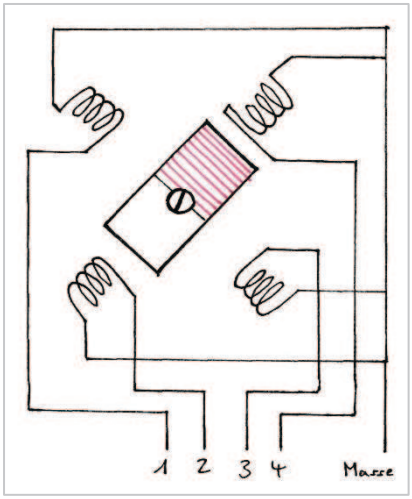


Abb.1

2.2. 4 m Elevation (Abb. 6)

Für die Elevation des großen Radioteleskops verwenden wir einen Linearantrieb (Abb. 3). Um die Schubstange in Bewegung zu versetzten muss eine Trapezspindel (Abb. 2) gedreht werden. Die Trapezspindel verfügt über einen Kupplungsanschluss für den ursprünglichen Antriebsmotor, die aus einer Tragplatte mit drei Stiften besteht, auf die der Motor mit seinem Getriebe aufgesteckt wird. Demnach wird eine Kupplung benötigt, um den Schrittmotor an die Stifte setzen zu können. Zusätzlich wird der Motor fixiert, da er sich weder im Gehäuse verdrehen noch in Längsrichtung verrutschen darf.

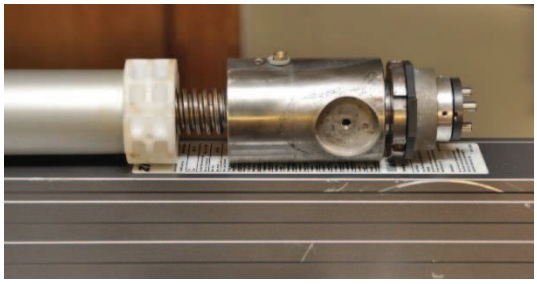


Abb. 2

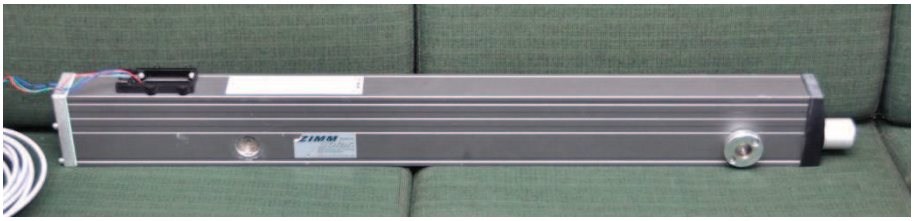


Abb. 3

Als Kupplung dient ein Nylonzylinder, in den vier Bohrungen eingelassen sind. In drei davon werden die Stifte, in die vierte über eine Madenschraube die Motorachse fixiert. Durch die Flexibilität des Materials kann man die Stifte und die Achse lückenlos in den Zylinder pressen. Zusätzlich gleicht die Kupplung Stöße aus.

Zum Schutz vor Verdrehung des Motors wird auf diesen eine Aluminiumplatte (Abb. 4) geschraubt. Da der Schrittmotor in das runde Gehäuse (Abb. 5) eingesetzt wird, lassen sich zwei hervorstehende Quadrate als Führung verwenden, die in die Schiene geschoben werden.

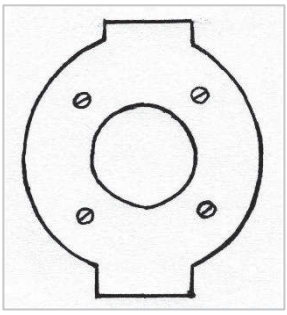


Abb. 4

Abb. 5

Um den Schrittmotor am Verrutschen in Längsrichtung zu hindert, wird von hinten ein weiterer Nylonzylinder herangesetzt. Dieser ist etwas kleiner und lässt sich so in die verschließende Platte einführen. Der Nylonzylinder kann so nicht seitlich verrutschen kann und dichtet das Gehäuse ab.

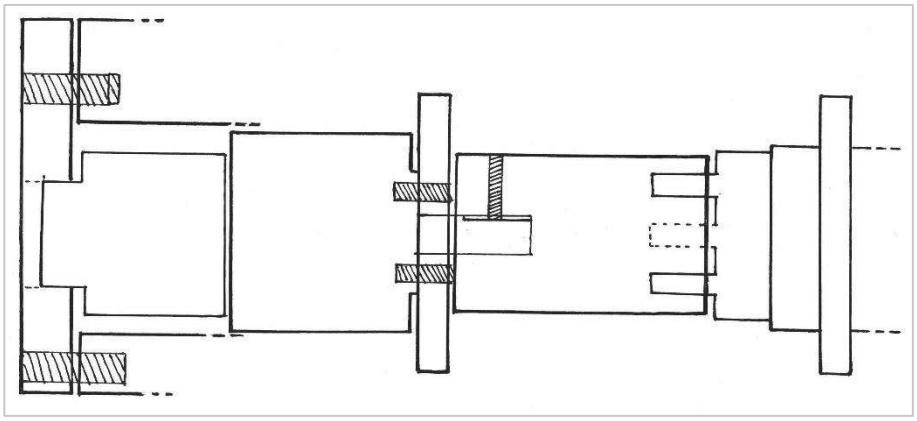


Abb. 6

2.3. 1,8 m Azimut (Abb. 9, 10)

Auch für den Azimut-Antrieb der 1,8 m Schüssel wird der Schrittmotor über eine Kupplung auf die Drehachse gesetzt. Die Befestigung des Motors am Tragdreieck (Abb. 7) muss verstellbar sein um ihn präzise über der Drehachse platzieren zu können.
Für diese Fixierung nutzen wir eine Schiene, die drehbar und verschiebbar am Tragdreieck befestigt wird. In die Schiene wird unten eine Platte geschoben, die ein Loch mit einem Gewinde hat. Durch eine Bohrung im Tragdreieck lässt sich die Schiene so am Gerüst befestigen.

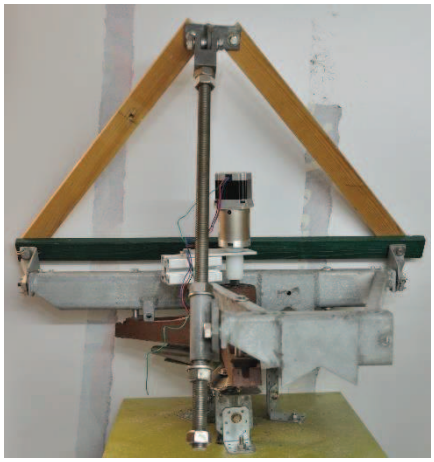


Abb. 7

Zur Befestigung des Motors an der Schiene wird an diesem eine Platte (Abb. 8.) mit zwei Bohrungen angebracht. Eine weitere kleine Platte, die oben in die Schiene geschoben wird, erhält zwei Bohrungen mit Gewinde, sodass die Platte mit dem Motor verschiebbar verbunden ist. Die hierbei verwendeten Aluminiumprofile sind ausreichend dimensioniert, um den wirkenden Hebel- und Torsionskräften standzuhalten.

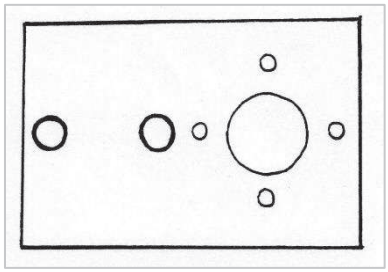


Abb. 8

Für die Kupplung verwenden wir abermals einen Nylonzylinder mit zwei grundlegenden Bohrungen (12 mm, 16 mm). Drei Gewinde Bohrungen fixieren mit Madenschrauben die 12 mm Motorachse in der Kupplung. Mit der 16 mm Bohrung wird die Nylonkupplung auf die gezahnte Mittelachse des Hauptlagers aufgepresst und zusätzlich mit einer weiteren Gewindebohrung mit Madenschraube fixiert.

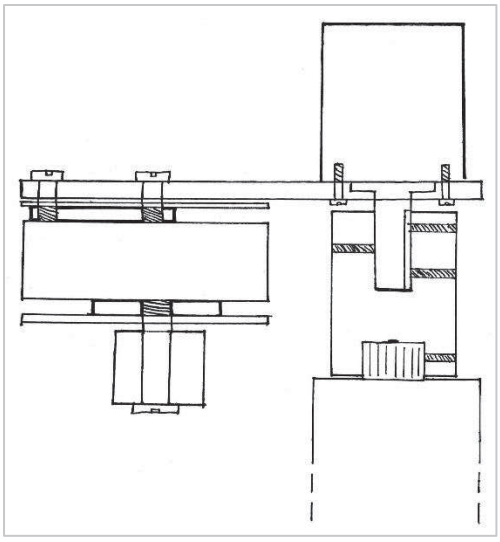


Abb. 9

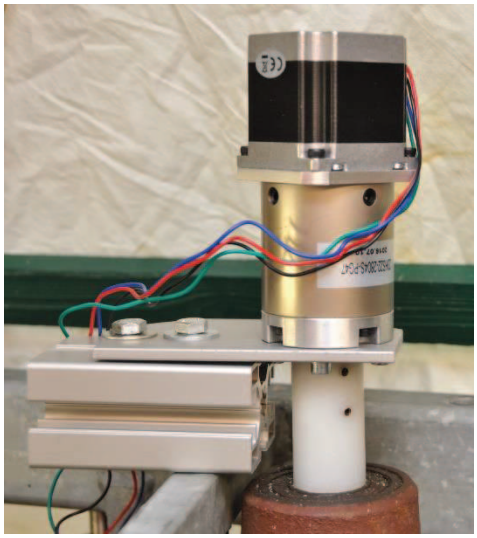


Abb. 10

2.4. 1,8 m Elevation

Um den Parabolspiegel der kleinen Schüssel bewegen zu können, muss genauso wie bei der großen eine Gewindestange gedreht werden. Daher haben wir uns für dasselbe Antriebssystem entschieden wie es bereits in 2.2 geschildert wurde. Auch hier soll ein Linearmotor verwendet werden, jedoch auf Grund der kleineren zu Bewegenden Masse der Nema 14.

3. ST Microcontroller 32 Nucleo Board

Der sogenannte STM, ein Microcontroller (Abb. 11) der vielseitig als Informationsvermittler eingesetzt werden kann, fungiert in unserem Aufbau als Blackbox und stellt innerhalb unseres Systems die Verbindung zwischen Programm und den Antriebssystemen dar. Die Programmierung des STM erfolgte als externe Hilfe durch Lukas Jürgens, ein ehemaliges Mitglied unsere AG. Die Basis-Platine wurde ergänzt durch Motortreiber, Display und eine Aufsteckplatine, auf der sich die zur Verwendung nötige Verkabelung befindet. Das gesamte Controllersystem wurde in ein wasserdichtes Gehäuse eingesetzt, da es später dauerhaft Teil des im Freien stehenden 4m Radioteleskopaufbaus sein soll und daher vor äußeren Einflüssen geschützt sein muss.

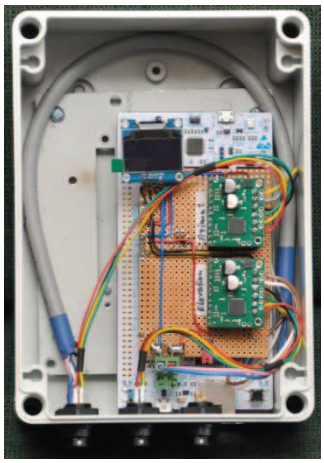


Abb. 11

Im Ansteuerprozess der drei in unserem Aufbau enthaltenen Motoren erhält der STM die Befehle unseres Programms und gibt sie an jene weiter. Die STM-Steuerung kann nur zwei Motoren gleichzeitig ansteuern, da diese aber durch ihren Steckeranschluss leicht austauschbar sind, lassen sich alle drei Motoren testen. Setzen sich die Motoren in Bewegung, reguliert der STM die Geschwindigkeit beim Anfahren langsam hoch, bis die vom Programm Angegebene erreicht ist und bremst die Motoren kurz vor Ende der auszuführenden Bewegung langsam bis zum Stillstand ab. Die dazugehörige Steuerkurve würde vom STM berechnet und berücksichtigt die Trägheit der zu bewegenden Masse, die zu verfahrende Strecke und die zulässige Höchstgeschwindigkeit. Denn entsprechenden Graphen zeigt Abb. 12. Durch diese Geschwindigkeitsregelung werden Schäden an den Motoren und ihren Halterungsvorrichtungen, sowie ruckartige Bewegungen des Radioteleskops vermieden.

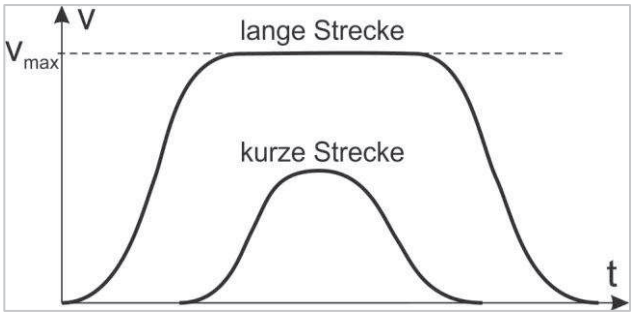


Abb. 12

Mit Hilfe von Steckleisten haben wir dem STM eine Lochplatine aufgesetzt. Die entsprechenden Kontakte wurden durch Kabel mit den auf der Platine platzierten Motortreibern und diese wiederrum mit den Motoren verbunden. Auch ein Display, für mögliche Fehlermeldungen ist an der Platine angebracht. Außerdem besitzt der STM einen Netzwerk- und einen Stromanschluss.

4. Programmierung

Das Ziel unseres Programmes sollte es sein, die präzise Ausrichtung unserer Radioteleskope mittels der hierfür montierten Schrittmotoren durchführen zu können, um dadurch umfassende Messungen mit diesen zu gewährleisten. Dazu zählt nicht nur das einfache Senden von Positionsbefehlen an den STM, sondern auch die Möglichkeit der sequenziellen Ausrichtung im Rahmen einer Messreihe festzulegen, sowie Arbeitsschritte beliebig oft zu Wiederholen oder voreingestellte Raster mit dem Teleskop abfahren zu können.
Wir entschieden uns die Programmiersprache C# (*„C- Sharp“*) zu nutzen. Nicht nur, dass diese dank ihrer Objektbezogenheit und seiner bereitgestellten Methoden und Befehle das Programmieren für Anfänger vereinfacht, sondern auch vor Allem, weil viele Projekte unserer AG bereits in C# geschrieben wurden, sie also als Standardprogrammiersprache bei uns gilt. So auch die bereits geschrieben Klasse „STM_CONTROL“ von Tim Rambousky, die uns die Kommunikation zwischen Programm und einem verbundenen Gerät ermöglicht.

4.1. Testprogramm

Um einen Überblick über den Umfang des Programmes zu erhalten, begannen wir damit, dessen Oberfläche zu gestalten (Abb. 13), um anschließend die einzelnen Funktionen zu programmieren. Dabei strukturierten wir die Schaltflächen, sowie die Ein- und Ausgabefenster in Gruppen. Insgesamt gestalteten wir 4 solcher Gruppen welche sich in Entwicklungsreihenfolge mit den Aufgabenbereichen Azimut-Motor, Elevations-Motor, Verbindungen und Übertragungsinformationen befassen. Im Folgenden möchten wir die genutzten Gruppen im Einzelnen erläutern. Hierbei sei angemerkt, dass bis zum heutigen Zeitpunkt noch nicht alle erstellten Schaltflächen mit Unterprogrammen (*Methode*) hinterlegt sind. Daher zunächst auch nur die Bezeichnung als „Testprogramm“.

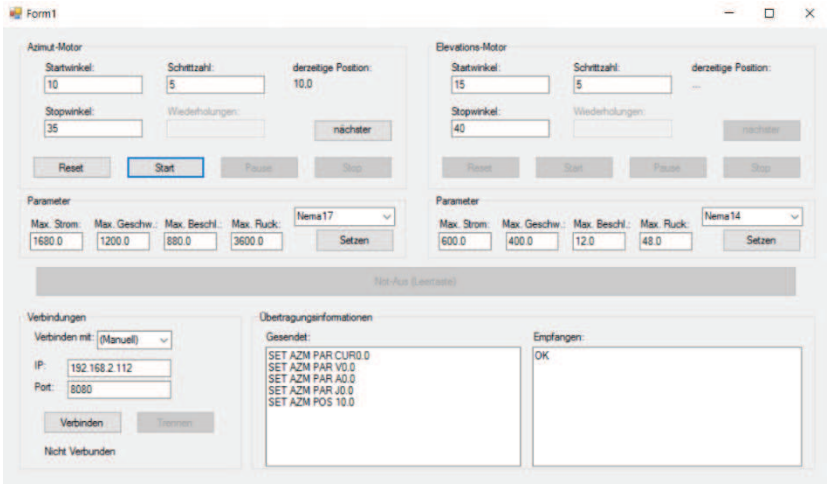


Abb. 13

4.1.1 Azimut-Motor (Abb. 14)

Wie der Name bereits vermuten lässt, haben wir in dieser Gruppierung alle Einstellungen aufgenommen die für die horizontale Ausrichtung des Teleskops notwendig sind. Hierzu zählen zum einen die Parameter für den entsprechenden Schrittmotor (Max. Stromstärke, max. Drehgeschwindigkeit, max. Drehbeschleunigung, max. Ruck) und zum anderen die originären Ausrichtungswerte für die Parabolantenne. Diese Ausrichtungswerte sind

Abb. 14

der Start- und Stopwinkel, also der Winkel, den der Motor zuerst anfahren soll, an dem die Messreihe gestartet wird, und der Winkel, bis zu dem das Radioteleskop abschließend gefahren werden soll, folglich wo die Messreihe endet. Die Schrittzahl bestimmt dabei die Anzahl der einzelnen Messungen über das eingegebene Winkelintervall. Mit diesen Informationen werden die benötigten Positionsbefehle berechnet und zum gewünschten Zeitpunkt dem verbundenen Gerät , im Normalfall dem STM, über die Methode „SendCmd“ übermittelt. Diese wiederum greift auf die Methode „SendTXT“ der Klasse „STM_CONTROL“ zurück.

Die Eingabe der gesamten Parameter sowie der anderen Werte ist nur innerhalb eines von uns festgelegten Toleranzbereiches möglich. Bei Abweichungen erscheint eine Fehlermeldung. Die Umsetzung erfolgt in dem zweigeteilten Unterprogramm „CheckValue“ bezüglich ganzer Zahlenwerte („int“, Abb. 15) und Dezimalwerte („float“). Alternativ können über ein Auswahlfeld vordefinierte Parametersätze für unterschiedliche Schrittmotoren abgerufen werden.

```
private int CheckValue(string text, int last, int min, int max, string message)
{
    bool ok = true;
    //Zuweisen "true"
    int wert = 0;
    try { wert = Convert.ToInt16(text); }
    catch { ok = false; }
    //Abfangen evt. Fehlermeldung/ Unterbrechung des Programms durch "falschen" Wert
    if (ok)
    {
        if ((wert < min) || (wert > max)) ok = false;
        //Zuweisen "false"
    }
    if (ok) return (wert);
    // falls "richtiger" Wert, diesen ausgeben
    else
    {
        MessageBox.Show(message);
        // Fehlermeldung in Messagebox
        return (last);
        // falls "falscher" Wert, vorherigen Wert ausgeben
    }
}
```

Abb. 15

Des Weiteren wird auf alle Eingaben mit der Methode „ChangeComma“ (Abb. 16) das Format „x.x“ übertragen, da beispielsweise eine alltagsübliche Kommazahl von dem STM nicht als Wert anerkannt wird.

```
private string ChangeComma(string text)
{
    int n = -1;
    n = text.IndexOf(",");
    if (n >= 0) text = text.Substring(0, n) + "." + text.Substring(n + 1);
    return (text);
}
```

Abb. 16

Sind alle Parameter eingegeben muss zunächst die Schaltfläche „Setzen“ betätigt werden, wodurch diese an den STM gesendet werden. Erst jetzt ist es möglich die Messreihe mit den Befehlen „Reset“, „Start“, „Pause“, „Stop“ und „nächster“ in ihrem Ablauf zu steuern. Zurzeit haben wir die Funktionen „Pause“ und „Stop“ noch nicht mit entsprechenden Programmroutinen unterlegt. Auf „Start“ wird als Click Event die Schrittweite berechnet und der Startwinkel an den STM übermittelt, so dass der

Schrittmotor in Bewegung gesetzt wird. „Reset“ lässt den Schrittmotor zur Kalibrierung auf die Position „0“ fahren. „Nächster“ dient als Zwischenlösung um das Radioteleskop im Rahmen einer Messreihe in die folgende Position gemäß berechneter Schrittweite zu bewegen, solange die noch nicht abgeschlossene Programmierung des STM einen automatischen Ablauf einer gesamten Messreihe nicht gewährleistet.

4.1.2 Elevations-Motor (Abb. 17)

Da sich Azimut- und Elevations-Antrieb in der Ansteuerung, bzw. den benötigten Funktionen zur Ansteuerung kaum unterscheiden, sind bis auf die Benennung und verschiedene Toleranzbereiche mancher Eingaben die Gruppen gleich aufgebaut.

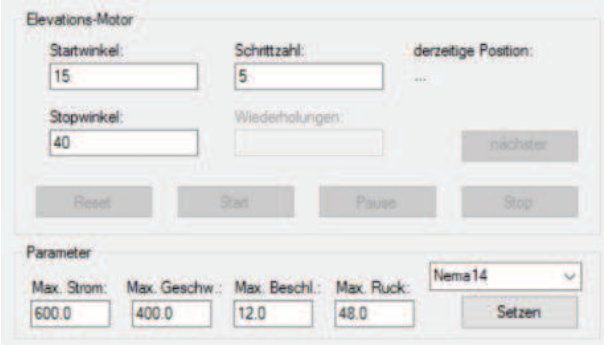


Abb. 17

4.1.3 Verbindungen (Abb. 18)

Unter der Gruppe „Verbindungen“ sind alle die Einstellung zusammengefasst, die wir vornehmen müssen, um eine Verbindung zwischen beispielsweise dem STM und dem Programm herstellen zu können. IP und Port des STMs oder unseres Servers sind bereits als Auswahlmöglichkeit gespeichert und lassen sich ähnlich wie bereits bei den Parametersätzen der Motoren über eine „Group- Box“ eingeben. Soll die Verbindung mit einem anderen Gerät eingegangen werden, können wir manuell jegliche beliebige Adresse eintragen. Voreingestellt sind die Daten für den STM, aus dem Grund, dass wir uns mit diesem vermutlich am häufigsten verbinden werden. Mit dem Betätigen der Schaltfläche „Verbinden“ geh das Programm nicht nur die Verbindung mit dem angegebenen Gerät ein, sondern beginnt über die Methode „StartListen“, die ebenfalls auf die Klasse „STM_CONTROLL“ zurückgreift, auch damit, auf Aktionen des anderen Gerätes zu achten. Sollte dieses versuchen Information zu übermitteln, werden jene erfasst und ausgegeben. War das Verbinden erfolgreich, ändert sich der momentane Status auf „Verbunden“ und das Element „Trennen“ wird entsperrt. Umgekehrt wechselt mit dem Drücken des „Trennen“ Buttons der Status entsprechen auf „Nicht Verbunden“ und die zuvor gesperrte Funktion Verbinden wird wieder verfügbar.

Abb. 18

4.1.4 Übertragungsinformationen (Abb. 19)

Die letzte Gruppe entwickelten wir vor allem, weil wir uns nur so einen Überblick über die gesendeten Befehle aber auch empfangenen Rückmeldungen schaffen konnten. Ursprünglich platzierten wir für jeden Motor ein Ausgabefeld, dass sowohl die übermittelten als auch erhaltenden Informationen bezüglich diesen Antriebs anzeigen sollte. Da der STM in seiner Antwort jedoch nicht zwischen Azimut- und Elevation unterscheidet, funktionierten wir sie als Textfelder um, die gesendete und empfangene Daten unabhängig vom Motor

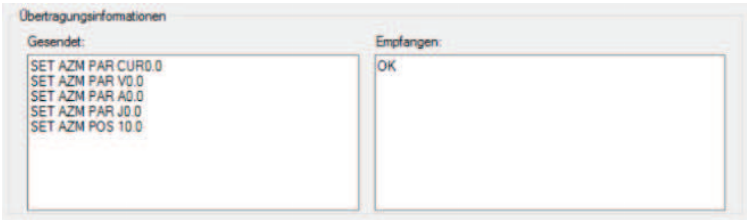


Abb. 19

darstellen. Das Fenster mit den ausgegangenen Informationen erhält dabei mit inbegriffen in die Methode „SendCmd“ die Anweisung, alle übermittelten Befehle zu dokumentieren. Eingehende Inhalte werden dem Unterprogramm „AcceptData“ (Abb.20) von der bereits in 4.1.3 erwähnten Programmstruktur „StartListen“ übergeben und schließlich über die Methode „WriteText“ (Abb. 20) in dem entsprechenden Textfeld ausgegeben.

```
private void AcceptData(object sender, NewDataEventArgs e)
{
    this.WriteText(e.Data);
}

delegate void WriteTextCallback(string text);
private void WriteText(string s)
{
    if (txtMessagesElv.InvokeRequired)
    {
        WriteTextCallback d = new WriteTextCallback(WriteText);
        this.Invoke(d, new object[] { s });
    }

    else
    {
        txtMessagesElv.Text += s + Environment.NewLine;
```

Abb. 20

4.2. Erprobung

Der erste praktische Test mit dem umgerüsteten Linearantrieb, um die Präzision und Reproduzierbarkeit zu testen, ist positiv ausgefallen. Es ist gelungen den Antrieb innerhalb von 35 s mehrmals um 500 mm zu verfahren und ohne erkennbaren Positionierungsfehler wieder in die Ausgangsposition zurückzuführen. Im Vergleich zu dem bisherigen Antrieb mit einem Gleichstrommotor hat sich dabei die Einstellzeit auf 1/20 reduziert (der Gleichstrommotor benötigte für die gleiche Strecke fast 10 Minuten = 600 s). Außerdem konnten wir, durch die erfolgreiche Ansteuerung des freistehenden Nema14-Motors zeigen, dass unser Aufbau funktionsfähig ist.

5. Ausblick

5.1. 1,8 m Elevation

Die Elevation des 1,8 m Reflektors ist zwar geplant, jedoch aus Zeitgründen noch nicht vollständig umgesetzt worden, weshalb dies unser nächster Schritt sein wird. Darauf soll eine Zusammenführung aller Projektkomponenten folgen, sodass es uns möglich ist weitere Präzisionsmessungen an den Radioteleskopen selbst durchzuführen.

5.2. Fehlende Programmfunktionen

Da der STM bis zum jetzigen Zeitpunkt nicht vollständig programmiert ist, war es uns noch nicht möglich alle vorgesehenen Funktionen der in 1.1 dargestellten Oberfläche umzusetzen. So erhalten wir beispielsweise zurzeit nur ein „OK“ als Rückmeldung dafür, dass ein vom Programm gesendeter Befehl empfangen wurde, nicht aber wann der Befehl ausgeführt wurde oder an welcher Position die Motoren sich gerade befinden. Es stehen uns folglich Informationen nicht zur Verfügung, die für eine Methode wie zum Beispiel das „Rasterfahren“ unumgänglich sind. Auch die Funktion einer durchgängigen Positionsrückgabe oder die Möglichkeit Befehle beliebig oft maschinell zu wiederholen, konnten wir daher bisher nicht realisieren. Darüber hinaus sind die Schaltflächen „Pause“, „Stop“ und „Not-Stop“ noch nicht mit Unterprogrammen hinterlegt. Es existieren zwar bereits Programmstrukturen oder „Prototypen“, die sich mit den genannten Teilen des Programms beschäftigen, jedoch noch auskommentiert, also inaktiv sind. Diese Unzugänglichkeiten werden wir schnellstmöglich nach Erreichen der vollen Funktionsfähigkeit des STM umsetzen.

6. Danksagung

Wir möchten uns zu aller erst bei unserem Betreuer Thomas Biedermann bedanken, der uns immer mit Ratschlägen zur Seite stand und sich auch außerhalb der AG-Stunden für uns Zeit genommen hat. Außerdem wollen wir uns bei Susanne Biedermann bedanken, die dafür sorgt, dass immer etwas da ist, an dem man sich stärken kann. Auch unsere Eltern möchten wir erwähnen, die immer wieder nach der AG die Autofahrt auf sich nehmen, um uns abzuholen.

7. Quellen

1. Nema14-Datenblatt: http://www.omc-stepperonline.com/download/pdf/14HS13-0804S-PG5.pdf
2. Nema17-Datenblatt: http://www.omc-stepperonline.com/download/pdf/17HS13-0404S-PG5.pdf
3. Nema23-Datenblatt: http://www.omc-stepperonline.com/download/pdf/23HS22-2804S-PG47.pdf
4. STM32-Nucleo Datenblatt und Benutzerhandbuch: http://www.st.com/content/ccc/resource/technical/document/user_manual/group0/26/49/90/2e/33/0d/4a/da/DM00244518/files/DM00244518.pdf/jcr:content/translations/en.DM00244518.pdf
5. Zimm-Linearantrieb: http://de.zimm-solar.com/downloads/datenblatt-zsa-21
6. TCP/IP-Anbindung: Tim Rambousky, „Strukturierte systemübergreifende Kommunikation mit TCP/IP“, Jugend-forscht-Projekt 201