AR2 Operating Instructions

Introduction

I chose to build Chris's AR2 because he made his maths available and was generous enough to provide a series of YouTube videos explaining in detail how the maths worked. I wrote the software and was sure it would work. But then I had to build the arm to prove it.

If anyone else can use my software please feel free to do so.

The real power in the software is the integration of the MS Scripting Engine. I prefer to use VB scripting as I'm familiar with it but you can use Jscript just as easily.

I have provided a number of methods and properties which are available in the script which provide a bridge between the sketch running on the microprocessor and the scripting engine which runs as an in-process server.

Caveats and Warnings

This is new software which works fine in my environment with my motors, my build idiosyncrasies, my microprocessor and my operating system. Yours will be different, so the outcome may be different. Be ready to turn off power to your arm if things go 'pear-shaped'.

Firstly, I am using the AR3 steppers and the stepper driver setup is slightly different to the AR2. This may explain why the stepper limits I have are different to Chris's defaults. If you are using an AR2, have a look at the constants **elbow_StepLimit** which you may like to change to 7850, and **wrist_yaw_StepLimit** to 15200. You find these at the top of the sketch.

My limit switches use only 2 wires. The common pin goes to ground and the NC pin goes to the Teensy configuration pin. Bounce handles everything from there.

Most of the sketch was written on an Arduino Mega until I discovered that a Teensy is not teensy at all but 10 times faster, 32-bit and with a floating point unit (albeit on the 3.5 it can't process doubles in hardware). I have not revisited the Mega with the latest software so I don't know if there are any issues but I think it will be OK.

Libraries

You will need to install the teensyduino library (https://www.pjrc.com/teensy/teensyduino.html) if you are using a Teensy (but you already knew that), the Matrix Maths library which is on GitHub (http://playground.arduino.cc/Code/MatrixMath) and the AccelStepper library, which I think I also found on GitHub. The AccelStepper library simplifies the motion profiling to an extraordinary extent and reduces the code considerably.

Bounce2 and Servo should already be installed.

Make sure your USB drivers are installed as well.

The Sketch

The sketch runs in a command/response mode. You can execute all the available commands from the command line. Just start a serial window from the sketch, type in a command string and press return. The command will be executed and the sketch will reply with ‘Pronto’, and is ready for the next command. I’ll list the commands later.

The forward and inverse kinematic routines are interpretations of Chris Annin’s Python code. All the matrices are 2-dimension arrays and the matrix operations use the Matrix Maths library. You can print out all the intermediate tables if the sketch is in **verbose** mode. This enables you to check everything against Chris’s kinematic model. This was how I debugged my code.

When you run the sketch it will initialise as if the arm is configured. If you do not power up your steppers you can enter a list of x, y, z, yaw, pitch, roll or J1-6 angles and you will get back lots of information about angles and steps, etc.

But you must configure before you try anything with the steppers powered up.

The Commands

- a – followed by list of 6 angles.eg. a0,-90,90,0,90,0
- p – followed by a list of 6 global positions.eg. p286,0,440,-90,180,-90
- d – a number of milliseconds.eg.d1000
- c – Configure. 0 = all, 1, 2, etc . For example c0
- k – Move to park position. Arm is vertical and power can be removed
- h – Move to home position, all square
- m – Motion. Set speed, acceleration
- s – Set Servo position 0 – 180.eg.s45
- b - Toggle verbose setting
- v – Display version

Configuration

You must configure the arm before use. With the sketch running, power up the steppers, enter the command **c0**

The process is slow. Each stepper is reversed until its limit switch is triggered (goes high). The sketch first displays the state of the configuration pins. They should all be ‘off’. If not check your wiring.

Each joint is configured separately from J1 to J6. I do this so that the gripper doesn’t get caught up in the arm.

Note: When you’ve finished using your arm, execute the Park (k) command. This will place the arm in a vertical position, including the gripper, so that the next configuration will work smoothly.

After the joint is configured, it will move to its home position. When all joints have been configured, the arm should be in the fully home position.

Home Pose

The home pose gives you an indication that the arm is correctly calibrated. The home co-ordinates are:

X=286, y=0, z=440, yaw=-90, pitch=180, roll=-90

… this results in the angles …

J1=0, j2=-90, j3=90, j4=0, j5=90, j6=0

The arm should be pointed forward, the shoulder should be vertical, the elbow horizontal, the wrist flat with the 'fingers' pointed down at 90 degrees. If it's not, play with the step limit constants at the beginning of the sketch.

You're good to go….

Command Application

This application is written in object Pascal in my Delphi environment. I have used Delphi commercially for decades and I have all the tools and code base to make my life easier. Too bad it's not such a popular development environment otherwise I'd release the source for this as well.

Make sure the USB drivers are installed.

The command application AR2SerialCommunication makes the operation of the arm easy. Once the script is running (no serial window please). Start the application. For the first run, from the File menu, select Connection Parameters and select the COM port. If you can't see it, you haven't installed the USB drivers. The connection parameters are stored in the associated ini file. You shouldn't have to change anything else.

I need some screen shots….

Open the serial connection by pressing Connect.

From the Configuration menu select All. Wait for the arm to configure. If it looks OK, you're ready to go.

The Raw Output shows all the returned messages from the microprocessor,

You can change the position co-ordinates and press the associated Send button or change the joint angles and press its associated Send button to jog the arm. You can save any sequence of position co-ordinates to the File display with the Save button.

From the File menu you can save and restore a file of position co-ordinates with Open and Save.

VB Scripts

From the **Scripts** menu, select **Load** to read a script. There is some error checking but you may get an error when you're running the script. Fix the error, save the file and re-load.

You can view the script with **Show** but you can't edit and save.

Select **Execute** and be prepared to halt the script if necessary. Use the **Stop File** button or pull the power.

I have provided a sample script ‘pickup and drop.vbs’. It simulates a simple loop where the pickup point get lower and the drop point gets higher.

Use Notepad++ to edit your scripts or any editor that understands VB syntax.

Here are the methods and the property I have provided to link the script with the sketch.

```
Sts = AR2Auto.Execute( x, y, z, yaw, pitch, roll, delay )
```

Pretty obvious what this does. It includes a delay (pretty common) which can be zero.

```
Sts = AR2Auto.Notify( "Starting script " & script_name )
```

Passes messages back to the Script Output tab

```
Sts = AR2Auto.Servo( 170 ) 'close
```

Manipulate the servo. It just takes a number between 0 and 180. You could pass 0 and 1 and use it as a Boolean. I can also add new methods if requested.

```
Sts = AR2Auto.Delay( 500 )
```

Pass a delay in milliseconds command to the sketch

```
 AR2Auto.Stop_script
```

This is a property that is set in AR2SerialCommunication. If you press **Stop File**, the script will stop. Note that once the script has stopped you need to press the button again to toggle it, or you script will stop immediately next time you try and run it.

Database Connectivity

Is anybody interested in connection to SQL Server and loading up scripts from a database?

HIM 28.May.2020

-oOo-