Radio Controlled SG5-UT Robotic Arm

One question that comes up often is “Can the SG5-UT be RC controlled?” So, I went into the lab and came up with a robust and expandable solution, RC_SG5ArmV1_0.bsp. This is a STAMP application that uses the following hardware and software.

Hardware

SG5-UT Robotic arm kit
Parallax Basic STAMP 2p
Parallax Board of Education
Parallax Servo Controller
Tower Hobbies System 3000 6 channel FM Radio Control System
Tower Hobbies System 3000 7 channel receiver

Software

Adjust_Bicep.bsp
ArmEngine_Basic.bsp
Five_Channel_RC.bsp
RC.bsp
RC_SG5ArmV1_0.bsp

Overview

The radio control code (RC_SG5ArmV1_0.bsp) is actually a combination of two files ArmEngine_Basic.bsp and Five_Channel_RC.bsp.

ArmEngine_Basic.bsp contains the basic building blocks to control the SG5-UT. It is designed so that developers can build on existing functional code and not reinvent the wheel. Not everyone wants only an RC arm. ArmEngine_Basic.bsp is responsible for moving each of the 6 servos that make up the arm joints. It also constrains the joints to work in a particular workspace. This allows the developer to control where the arm does its work. The constraints can also be dynamically assigned during program execution.

Five_Channel_RC.bsp is the RC code portion. It is responsible for reading 5 receiver channels, determining if the transmitter and/or receiver are power and sending a signal, converting the receiver output to degrees, determining what joint(s) to move, and error checking. You can also use Five_Channel_RC.bsp to troubleshoot your receiver connections.

All 3 files mentioned have a troubleshooting section and are thoroughly commented. Troubleshooting is a simple procedure that writes all the variables used in the applications to a debug screen. This is a handy tool if you are making changes to the code or if you are experiencing problems. You will need to have some understanding of reading and writing PBasic code.

Setting up your SG5-UT Robotic Arm

That's enough overview, let's get started. I recommend following the steps outlined below before running RC_SG5ArmV1_0.bsp.

1. Check your PSC to SG5-UT connections. This is extremely important because this code uses connections that are different than those in the current assembly guide version 3.1, sg5ut_v3_1.pdf.

```
Base servo..........PSC Channel 0
Bicep servo.........PSC Channel 1
Elbow servo.........PSC Channel 2
Wrist servo.........PSC Channel 3
Empty servo.........PSC Channel 4 --> Place holder
Gripper servo.......PSC Channel 5
RightBicep servo.....PSC Channel 6
```

You might be wondering why channel 4 is empty. Channel 4 is dedicated for a rotating wrist control. The rotating wrist feature will be in our next SG series product release.

2. Chapter #5 "Adjusting the Biceps Servos" (sg5ut_v3_1.pdf) describes how to physically and programmatically align the bicep servos. It also asks you to write down the difference in servo positions values as seen by the PSC. This procedure has been modified. The Adjust_Bicept.bsp file allows you to programmatically align the bicep servos. It also displays a "RightBicepOffset Constant Declaration" that you should copy and use in all arm code going forward. I imagine that if you are reading this you might have already assembled the arm. You could use the value that you already have or you could align the arm with the new code. To realign you will need to remove a bracket and detach the elbow – sorry. I strongly recommend realignment though! I designed the code so that YOU MUST enter the "RightBicepOffset Constant Declaration" or the code will not run.

3. Read the comments in ArmEngine_Basic.bsp. There are a lot of comments, but they explain how the code functions. Run the code and verify that you can move all 5 joints. Below is a code sample form ArmEngine_Basic.bsp.

```
Main:
  array(Base) = 90
  array(Bicep) = 100
  array(Elbow) = 90
  array(Wrist) = 90
  array(4) = 0
  array(Gripper) = 120
  ctrlByte = %00000000        ' Reset control byte

  baseBit = 1
  biceptBit = 1
  elbowBit = 1
  wristBit = 1
  gripperBit = 1
```

```
  GOTO  Move_Arm_Joints
END
```

Notice the array values? If you want to verify that the base is connected and working set array(base) = 30 and run the code. The base should move. If not, you need to check your connections.

4. Read the comments in Five_Channel_RC.bsp. Again, there is a lot to read but it will help you to understand how the program works and how to connect the STAMP to a receiver. This code has debug mode turned on by default so that you can verify the STAMP is receiving a signal. Try running RC.bsp if you are having problems with Five_Channel_RC.bsp. RC.bsp simply displays raw PULSIN values on STAMP pins 10 through 14.

5. Last step is to run RC_SG5ArmV1_0.bsp.

Receiver Controls
It takes a few minutes to get used to the receiver controls. Use the table below for transmitter to joint relationships.

Receiver	Joint	STAMP	PSC
Throttle	Bicep and Elbow	Pin 12	Channel 1 ,2 & 6
Rudder	Gripper	Pin 13	Channel 5
Elevator	Wrist	Pin 11	Channel 3
Aileron	Base	Pin 14	Channel 0
Flaps	Elbow	Pin 10	Channel 2

If the transmitter controls are centered then nothing will happen. To move a SG5-UT joint move the related transmitter control. To stop a joint move the related transmitter control back to center. You can move multiple controls at one time.

That's about it. If you run into problems please visit CrustCrawler's robotics forms. Go to "Projects" -- "Radio Controlled SG5-UT Robotic Arm" then post a reply.

Happy Programming
Mike Gebhard