# Final Design Project: SolarRAT

CPE 233

Instructor: Dr. Bridget Benson

March 8, 2019

Partner 1: Julio Tena

Partner 2: Victor Delaplaine

Partner 3: Javier Flores

## Introduction

This project consisted of a solar panel that was controlled by the RAT MCU designed in class. A small solar panel was meant to do a 180-degree sweep, via a 9g servo motor, every few minutes to find the position corresponding to the highest source of light. In order to rotate the servo, 8 wires were connected to an Arduino Uno, which represented the signals coming from the Basys3 board. The signals produced in the RAT MCU were programmed to change based on an external input. For this particular project, the XADC ports from the Basys3 were used to read a voltage differential coming from a circuit that relied on a photoresistor as the variable input. Since the XADC has a rating of 0 - 1 V input, the circuit was designed to have the highest voltage correspond the the position with the most light intensity and the lowest voltage represent the location with the least amount of light. Through it all, at its core, this project takes advantage of the versatility of the RAT MCU in order to control inputs and outputs, as well as setting different modes of operation.

## Operation Manual

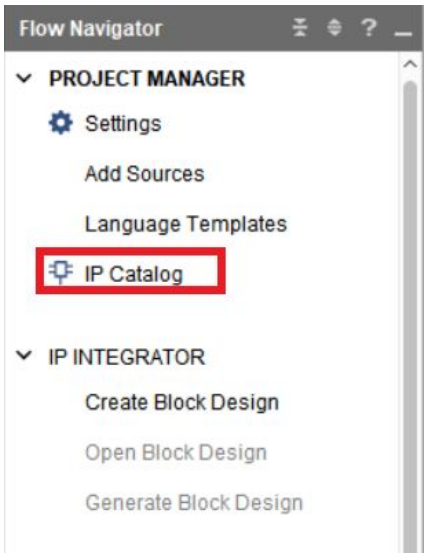
The SolarRAT was designed with two main modes of operation in mind. The most trivial is the automatic sweep which enables the user to simply wire the circuit correctly as shown in *Figure 1* and to program both the Basys3 and the Arduino with the code provided. Once the code is uploaded, the servo motor will begin to sweep as instructed via the RAT assembler code. At this point, the user is not required to interact with the Basys3 board since the solar panel will be oriented to its best location for maximum power transfer. However, if the user decides to manually position the solar panel in other locations, then the interrupt button (BTNL) on the Basys3 board must be pressed and the switch 7 must be toggled high to enter manual mode. Once in the manual mode, the solar panel can rotate at command with different combinations of the lower to switches. It is important to assert that the switch 7 remains high in order for the manual mode to be enabled, otherwise the sweep mode will resume. While at manual mode, leaving both lower switches off, will position the solar panel at 45°. When switch 1 is on and switch 2 is off, the servo will rotate 90°. With switch 1 off and switch 2 high, the servo will rotate 135°. When both switches are on, the servo will begin a sweep from 15° to 165° in increments of 15° and back from 165° to 15°. All of the positions are

assumed to be relative to ground. Finally, if the user decides to start all over, there is no need to reprogram the Basys3 board since there is a way to reset the program. The latter is accomplished by pressing BTNR on the Basys3 board.

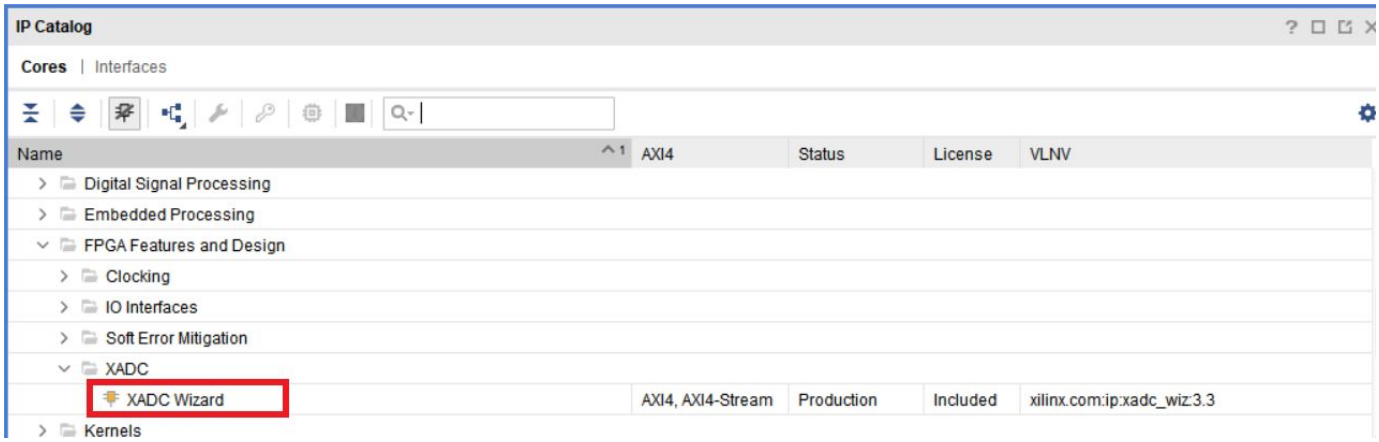
## Peripheral Details

The XADC ports required a special implementation of the module that was done with the help of a wizard in the Vivado software. A short explanation is described below. The following guide works with Vivado 2018.2. Other version may have different ways of enabling the XADC ports.

1. In the Flow Navigator Window located in the left side of the code, locate and click on IP Catalog

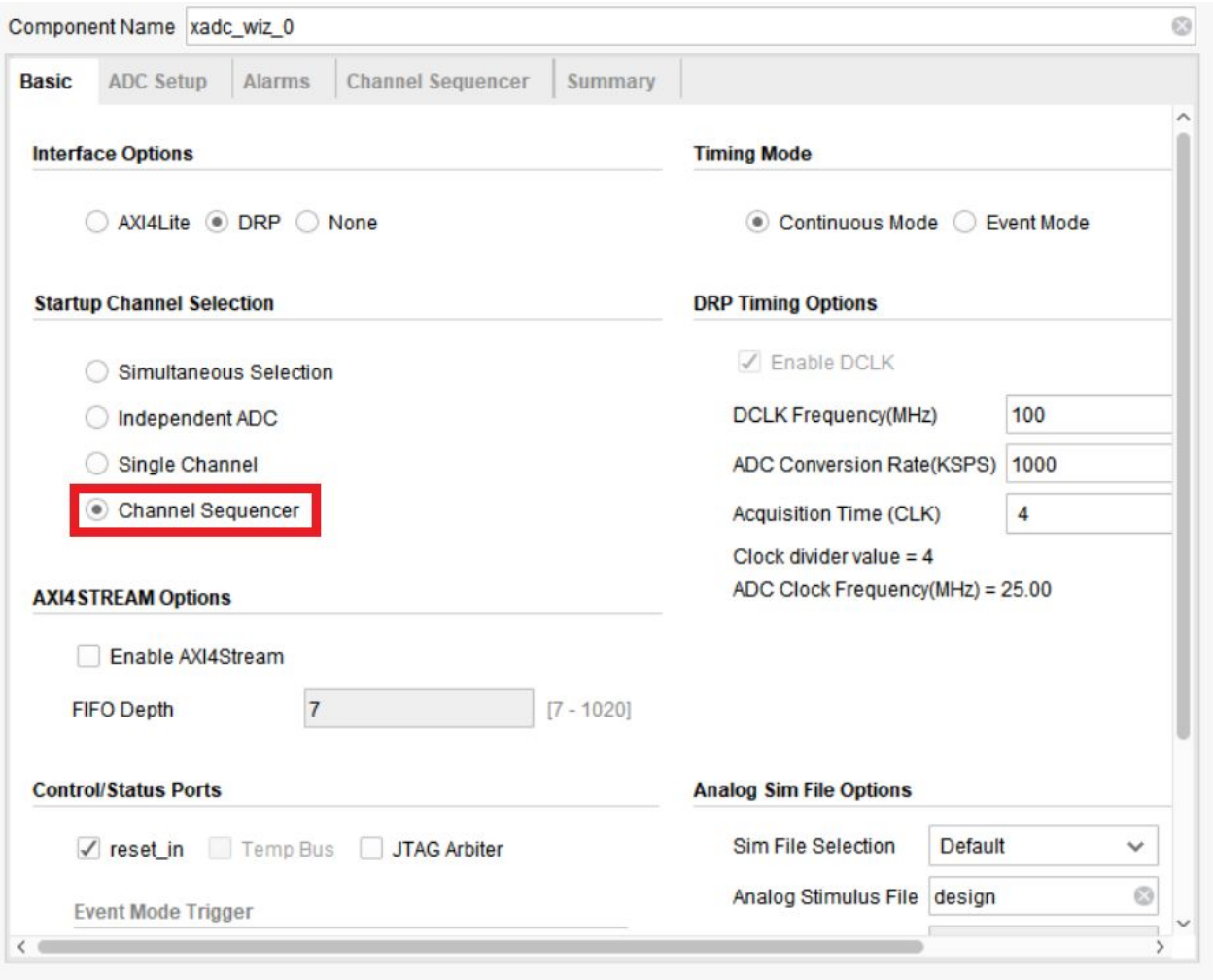


2. Scroll to find the FPGA Features and Design folder and expand. Select XADC → XADC Wizard



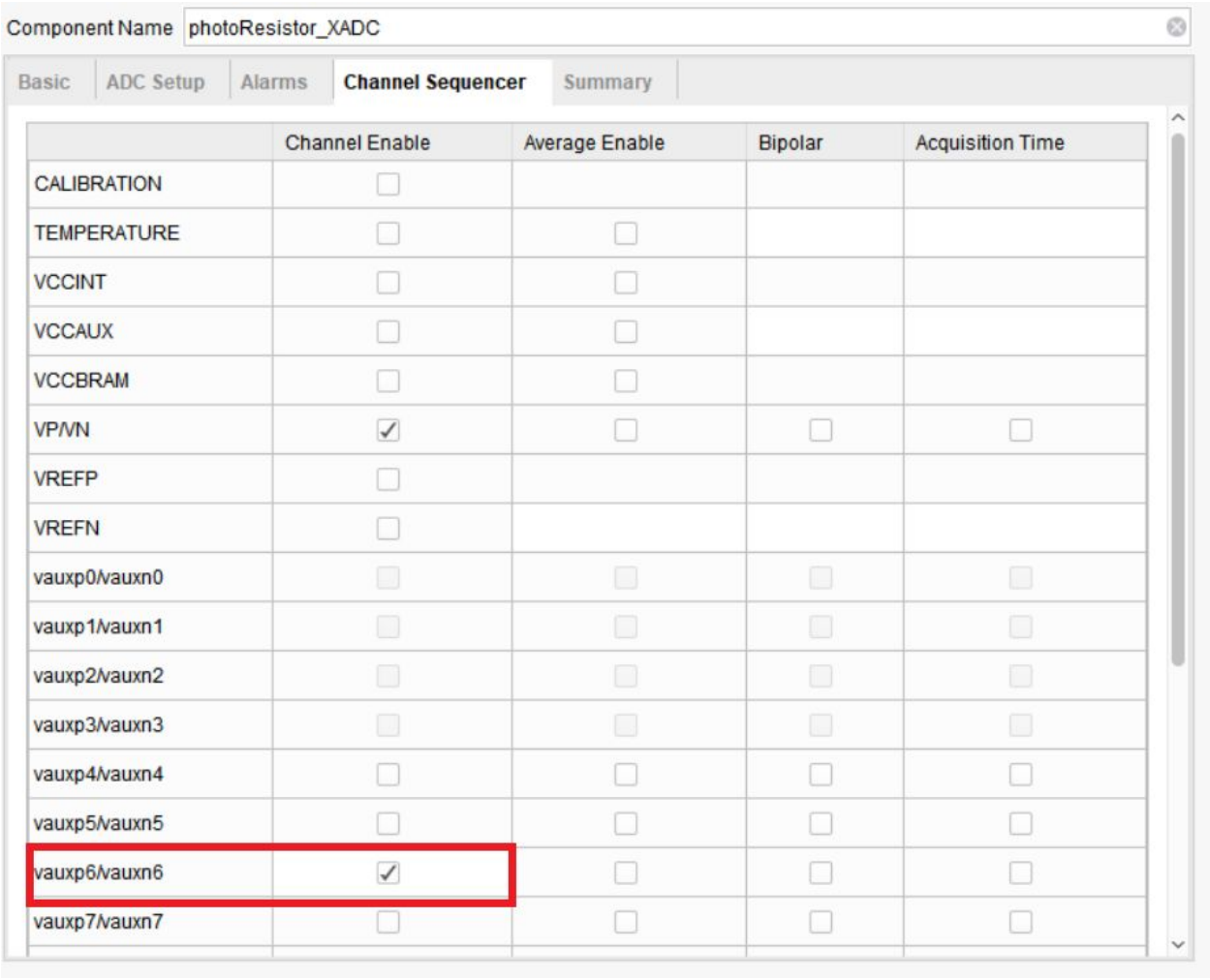

3. Under the Basic tab, select Channel Sequencer and select the Channel Sequencer tab.

    Note: At the top, the Component Name can be edited. This is the same as naming a module.




4. Select the appropriate channel-pair to enable. More than one can be enabled.

Note: This example uses only 1 channel-pair




5. Finish by pressing OK at bottom right corner.
6. The module can now be instantiated with the rest of your project. For more information see Appendix.

## External Circuit Peripheral

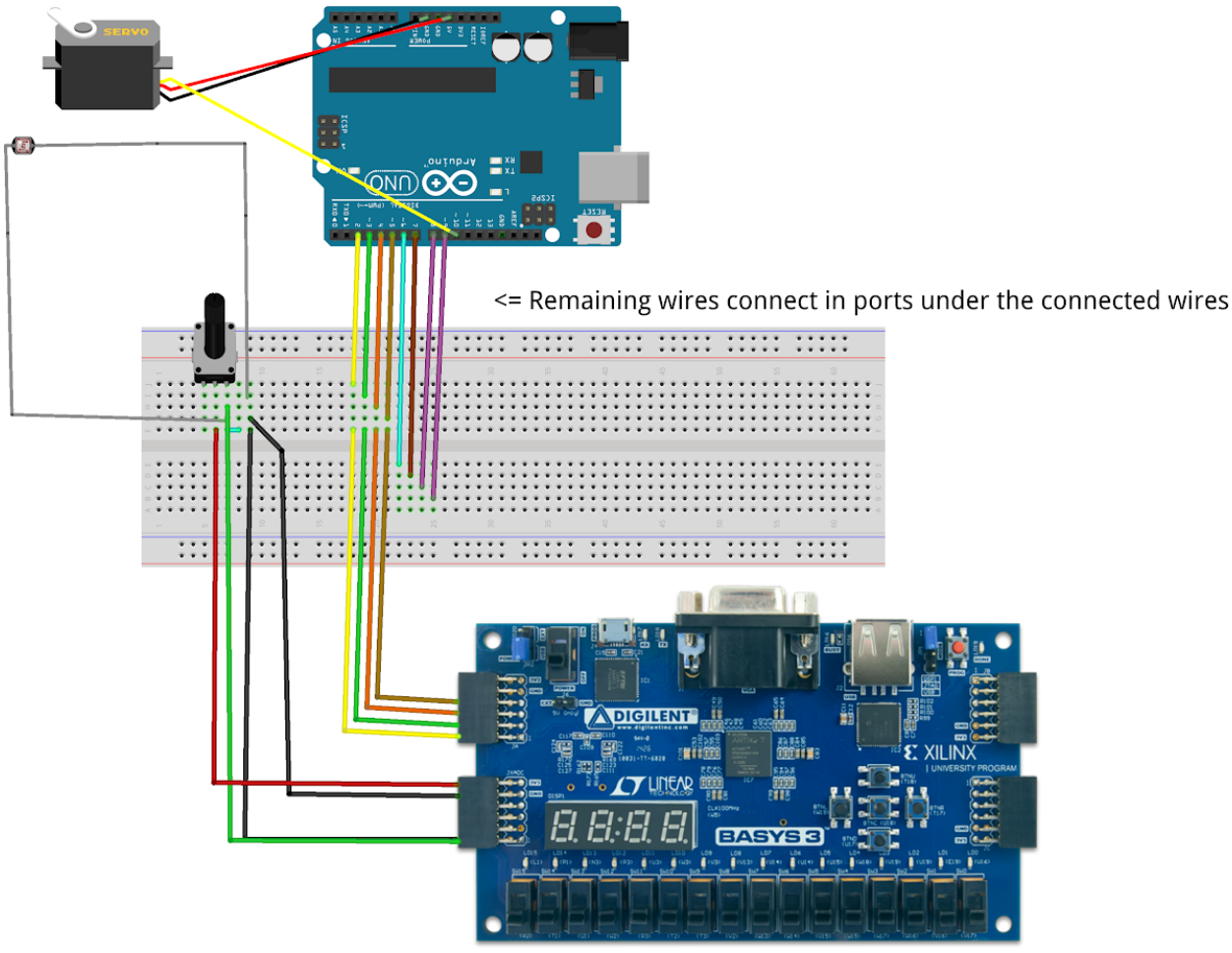



*Figure 1: Basys to Arduino Port Connections*

## Software Design

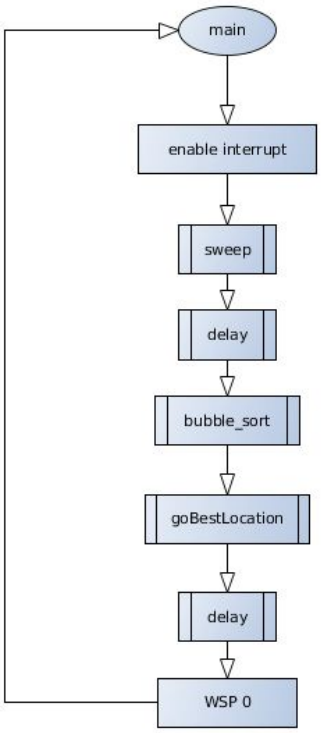
**Table 1 - Output of solarRat driver mapped into arduino**

| output of RAT Wrapper | Servo Rotation(Degrees) |
|---|---|
| 0000 | 0° |
| 0001 | 15° |
| 0010 | 30° |
| 0011 | 45° |
| 0100 | 60° |
| 0101 | 75° |
| 0110 | 90° |
| 0111 | 105° |
| 1000 | 120° |
| 1001 | 135° |
| 1010 | 150° |
| 1011 | 165° |
| 1100 | 180° |

## Table 2 - isr (manual mode)

| output[7:0](from basys3) | Servo Rotation(Degrees) |
|---|---|
| 100000_00 | 15° |
| 100000_01 | 75° |
| 100000_10 | 135° |
| 100000_11 | sweep(no data collection) |
| 000000_00 | sweep |
| 000000_01 | sweep |
| 000000_10 | sweep |
| 000000_11 | sweep |

- Note that output[7] tells the arduino that we currently in manual mode.



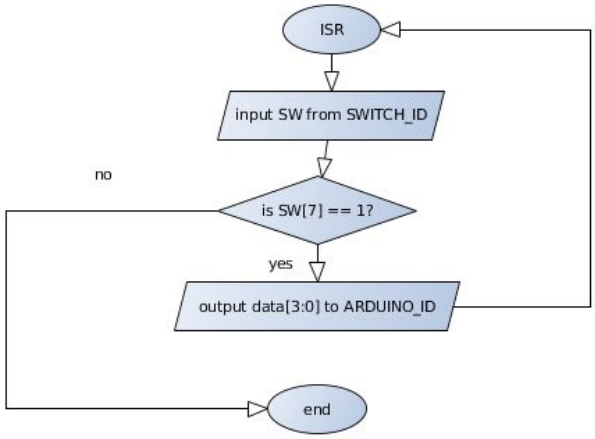



**Figure 2: main program flowchart beside shows the isr**

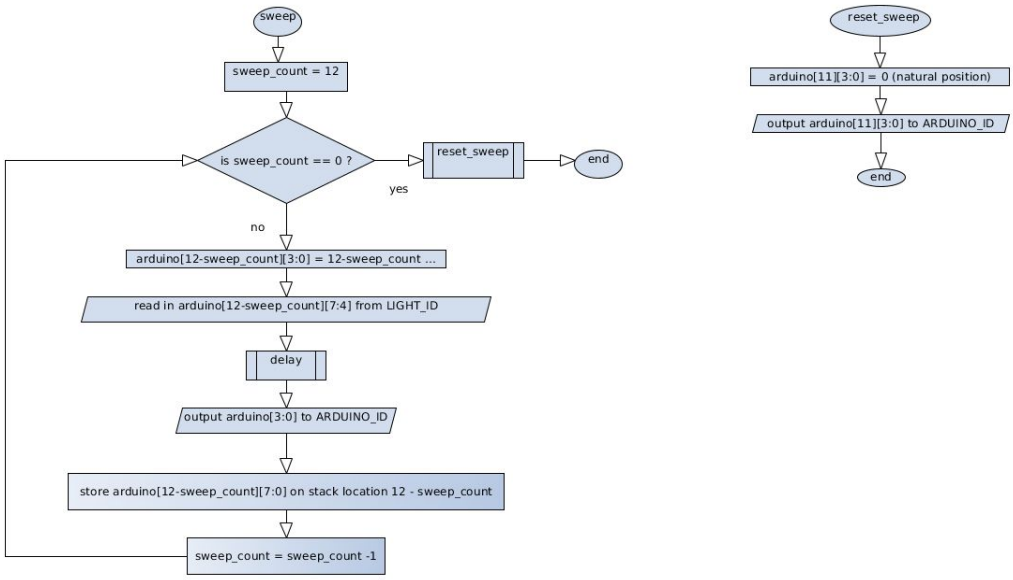



**Figure 3: sweep flowchart**

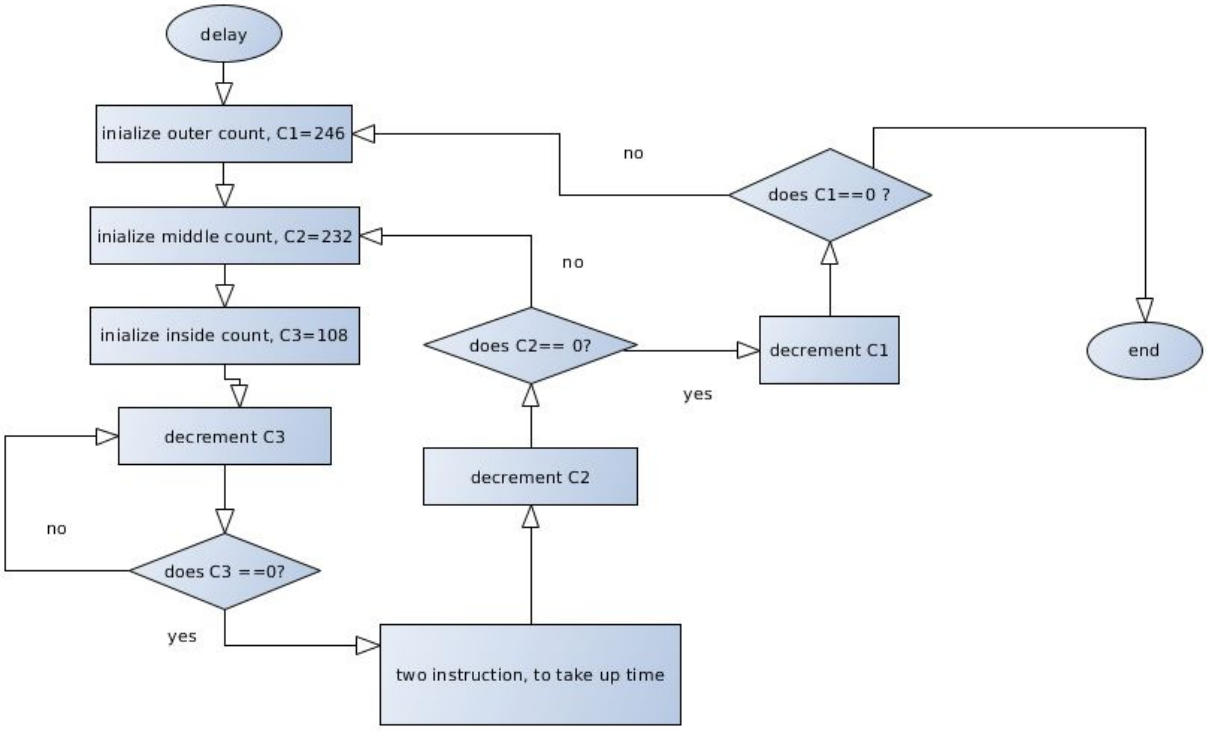



**Figure 4: Delay Flowchart**

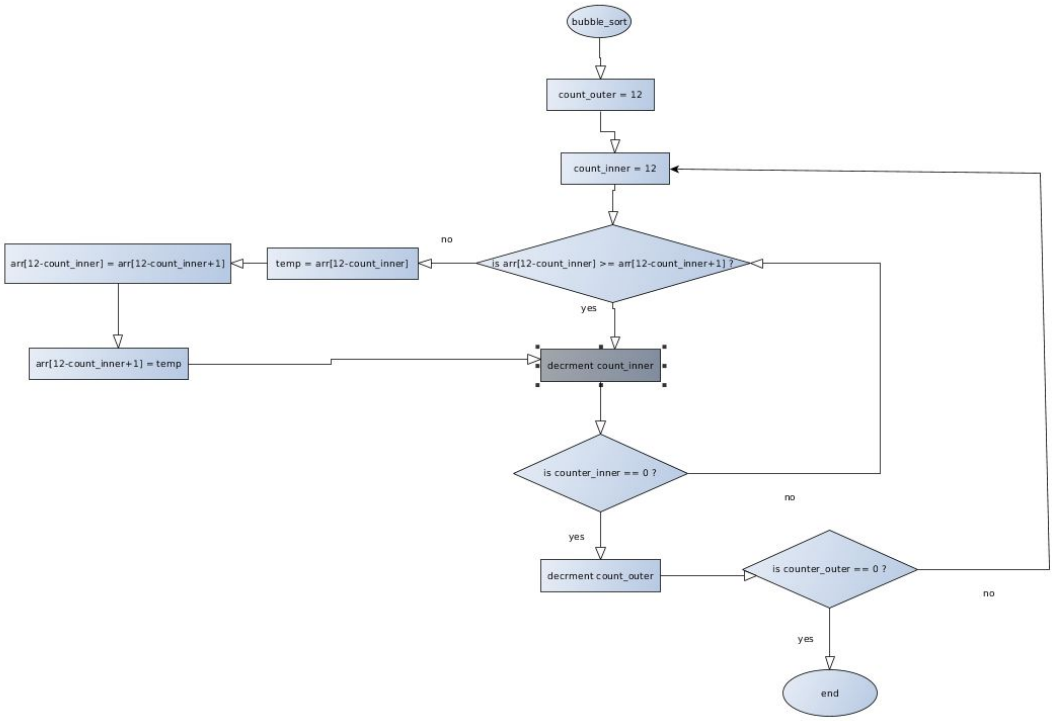



**Figure 5: Bubble sort Flowchart**

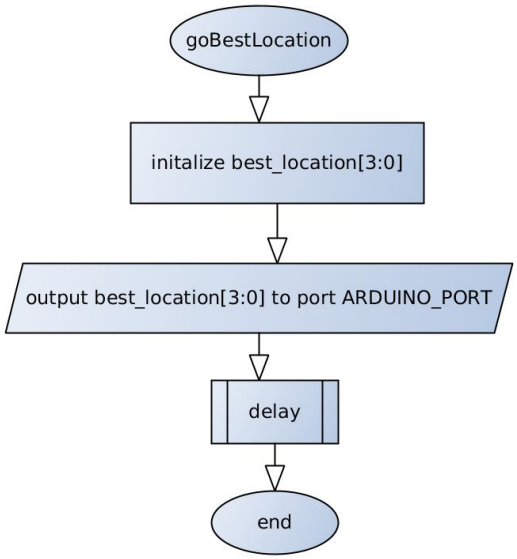


**Figure 6: goBestLocation Flowchart**

The RAT MCU enables interrupts to happen, which lets there be a manual mode in the Basys3 board. When the sweep is in happening it stores the values read at each position to the scratch ram. The best location will then get popped from the scratch ram and the solar panel will return to that position. If an interrupt occurs the code goes to an ISR state that lets switches 0 and 1 position the solar panel.

## RAT MCU ASSEMBLY CODE

```
;*******************************************************************************
; Final Project: main.asm
; Author: Victor Delaplaine and Julio Tena
; Date: 2019-03-09
; Description :
;Using A RAT MCU Created in CPE233, to input a set of data from a photoresistor using a
;servo to go through different set of locations (defined as a sweep subroutine). These
;locations (0-12 [4-bits] shown below in table 1) and the corresponding photon
;intensity (0-1V [4-bits]) are concatenated into one 8-bit register to be stored in the
;scratch ram of the RAT MCU. This sweep repeats until it goes from location 0-12, then
;a bubble sort algorithm is then called. This bubble sort sorts the highest
;Register[7:4] value on the top os the stack(TOS). After a goToBestLocation is called
;to pop the highest voltage value of the stack and is outputted to the
;servo_driver(arduino) to give the servo instructions to stay there for a longer time
;than usual.
;
; Register Uses:
; R0 - arduino[i][7:0]
```

```
; R1 - arduino[i][3:0]

; R2 - arduino[i][7:4]

; R3 - Variable for sweep_count

; R4 - Variable for 12 - sweep_count

; R5 - address of ith sweep position for arduino[i][7:0]

; R6 - outer count variable  (C1)

; R7 - middle count variable  (C2)

; R8 - inner count variable  (C3)

; R9 - outer count

; R10 - inner count

; R11 - ADDR for inner count arr[i]

; R12 - ADDR for inner count + 1 ( arr[i+1])

; R13 - temp

; R14  - used for input and outputting values (X)

; R15 - value fro arr[i]

; R16 - value for arr[i+1]

; R17 - best location[3:0]

; R18 - {ISR MODE ,0,0,0,0,0,SW[1:0]}

; R31 - using for zero

;---------------------------------------------------------------------



;-----------------Data Segment-----------------------------------
```

```
.DSEG
.ORG 0x01;
arduino_sweep: .BYTE 12 ; 0x01 ... 0x0C (12th)
;---------------------------------------------------------------------



;-----PORT DECLARATIONS----------------------
.EQU LIGHT_PORT  =  0x96
.EQU ARDUINO_PORT = 0x69
.EQU SWITCH_PORT = 0xFF
;-----------------------------------------



;----CONSTANT DECLARATION-------------------
.EQU DELAY_COUNT_INNER = 236
.EQU DELAY_COUNT_MIDDLE = 176
.EQU DELAY_COUNT_OUTER = 201
.EQU BUBBLE_OUTER_COUNT =  13 ;
.EQU BUBBLE_INNER_COUNT = 13
.EQU SWEEP_COUNT = 13
;-------------------------------------------------
.CSEG
```

```
.ORG 0x0D




main:
        SEI ; set interrupts
        CALL sweep
        CALL delay
        CALL bubble_sort
        CALL goBestLocation
        CALL delay
        WSP R31 ; have stack pointer go back to 0
        BRN main



;-------------------------------------------------------------------------------
; sweep  subroutine - goes through 15 degrees every 2s collects data and moves servo
;
; Tweaked Parameters :
;
; R0 - arduino[i][7:0]
; R1 - arduino[i][3:0]
```

```
; R2 - arduino[i][7:4]

; R3 - Variable for sweep_count

; R4 - Variable for 12 - sweep_count

; R5 - address of ith sweep position for arduino[i][7:0]

;

; Return : Nothing

;

;---------------------------------------------------------------------

sweep:

        MOV R3, SWEEP_COUNT ;sweep_count = 12

        MOV R4, SWEEP_COUNT

sweep_loop:

        CMP R3, 0 ; is sweep_count == 0?

        BREQ reset_sweep ; if yes == > PC = reset_sweep

        ;else

       MOV R4, SWEEP_COUNT ;



        SUB R4, R3 ; R4 = 12 - sweep_count

       ;(R4 == the location in which the motor is currently at)

        MOV R1, R4 ; arduino[3:0] = 12-sweep count

        IN R2, LIGHT_PORT ; arduino[7:4] = from LIGHT_PORT
```

```
        CALL delay
        OUT R1, ARDUINO_PORT ; output arduino[3:0] to Arduino_ID
        OR R1, R2 ; arduino[7:0]  = {arduino[7:4],arduino[3:0]}
        MOV R0, R1 ;
        ; before storing arduino[7:0] got to concatenate its componets
        ST R0, (R4) ; SCR[12 - sweep_count] = arduino[7:0]
        SUB R3, 1 ; sweep_count = sweep_count - 1
        BRN sweep_loop
reset_sweep:
        MOV R1, 0
        OUT R1, ARDUINO_PORT
        RET



;-------------------------------------------------------------------------------
; delay subroutine
; Delays for a given input of paramets R6, R7, R8
; Parameters : R6 - outer count variable  (C1)
;              R7 - middle count variable  (C2)
;               R8 - inner count variable  (C3)
;
;
```

```
; Return : Nothing

; Tweaked Parmeter : R6,R7,R8

;

;--------------------------------------------------------------------

;N_inner = 6

;N_middle = 4

;N_outer = 10

;3 inner

;2 middle

;1 outer

delay: MOV R6, DELAY_COUNT_OUTER ; R1 = BUBBLE_OUTER_COUNT



outer_loop:    MOV R7, DELAY_COUNT_MIDDLE

middle_loop:   MOV R8, DELAY_COUNT_INNER

inner_loop:    SUB R8, 1

               CLC

               CLC

               CLC

               CLC

               BRNE inner_loop
```

```
                CLC

                CLC

                BRNE middle_loop


                CLC

                CLC

                CLC

                CLC

                CLC

                CLC

                CLC

                CLC

                BRNE outer_loop


return:         RET

;-------------------------------------------------------------




;-------------------------------------------------------------------

; bubble_sort - subroutine returns highest voltage data on at SCR[0]
```

```
;
; Tweaked parameters
; R9 - outer count
; R10 - inner count
; R11 - ADDR for inner count arr[i]
; R12 - ADDR for inner count + 1 ( arr[i+1])
; R13 - temp
; R14  - used for input and outputting values (X)
; R15 - value from arr[i]
; R16 - value for arr[i+1]
;-------------------------------------------------------------------
bubble_sort:
        MOV R9, BUBBLE_OUTER_COUNT ; outer_count = 10
bubble_outer_loop:
        MOV R10, BUBBLE_INNER_COUNT ; inner_count = 3
bubble_inner_loop:
        ;get index of k and k+1 from 0->
        ;ADDR_i
        MOV R11, BUBBLE_INNER_COUNT
        ; ADDR_i+1
        MOV R12, BUBBLE_INNER_COUNT
        ADD R12, 1
```

```
        SUB R11, R10 ; ADDR_i = 5 - count_inner
        SUB R12, R10 ; ADDR_i+1 = 6 - count_inner
        ;get those values stored in the address
        LD R15, (R11) ; Y = arr[ADDR_i]
        LD R16, (R12) ; Z = arr[ADDR_i+1]
        ;COMPARE now
        CMP R15, R16 ;
      ;if we get c = 1 that means arr[ADDR_i+1] is greater than or equal to arr[ADDR_i]
      ;so lets swap if this is the case
        BRCS swap


decrement_count_inner:
        SUB R10, 1 ; count_inner = count_inner -1
        BRNE bubble_inner_loop
        SUB R9, 1 ;count_outer = count_outer - 1
        BRNE bubble_outer_loop
        RET


swap:   ;swap(arr[ADD_i], arr[ADD_i+1])
        MOV R13, R15 ; temp = arr[ADDR_i]
        MOV R15, R16 ; arr[ADDR_i] = arr[ADDR_i+1]
```

```
        MOV R16, R13 ; arr[ADDR_i+1]  = temp
        ST R16, (R12)
        ST R15, (R11)
        BRN decrement_count_inner




;-------------------------------------------------------------------------
; goBestLocation - takes a value arr[0] top of stack and goes to that position
;
; Tweaked parameters:
; R17 - best location[3:0]
; R31 - using for zero
;-------------------------------------------------------------------


goBestLocation:
               WSP R31 ; reg that has value of 0
               POP R17
               OUT R17, ARDUINO_PORT
```

```
                CALL delay
                RET


;-----------------------------------------


;-------------------------------------------------------------------------
; ISR - allows someone to go in manual mode, turn servo using SW's 45 degrees each
;
```

## Appendix

3D joints - https://www.thingiverse.com/thing:2271734

Stepper-to-PVC 3D-Part - https://www.thingiverse.com/thing:53321

Servo Motor -

http://www.ee.ic.ac.uk/pcheung/teaching/de1_ee/stores/sg90_datasheet.pdf

3D Printing Services on Campus - https://www.theinnovationsandbox.com/

BASYS3 XADC Demo-

https://github.com/Digilent/Basys-3-XADC?_ga=2.28328696.598648131.1551755682-1318774948.1543268595)

7 Series FPGA XADC -

https://www.xilinx.com/support/documentation/user_guides/ug480_7Series_XADC.pdf?_ga=2.35717988.598648131.1551755682-1318774948.1543268595