**Exercise 1: Change servo motor resolution (step size)**

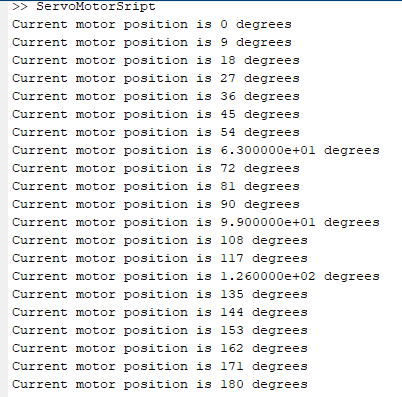
The shaft position is mapped from 0-180 degree to variable angle range 0-1 values. What is the angle (degree) represented by angle step size 0.05 ?

Modify the code as

angle = 0:0.05:1

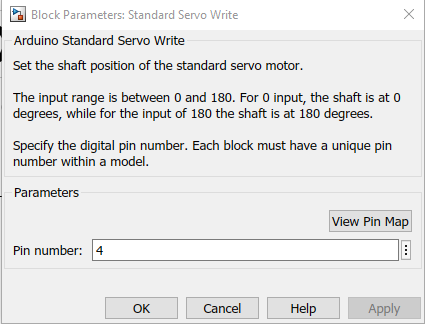
Attach the screenshot of motor position sequence.

Answer: The resolution angle is 180\*0.05 = 9 degree represented by step size 0.05.



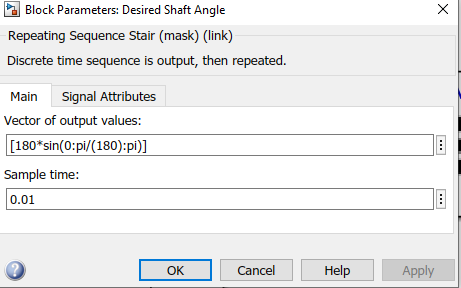
**Exercise 2: generating a repeated sequence stairs**

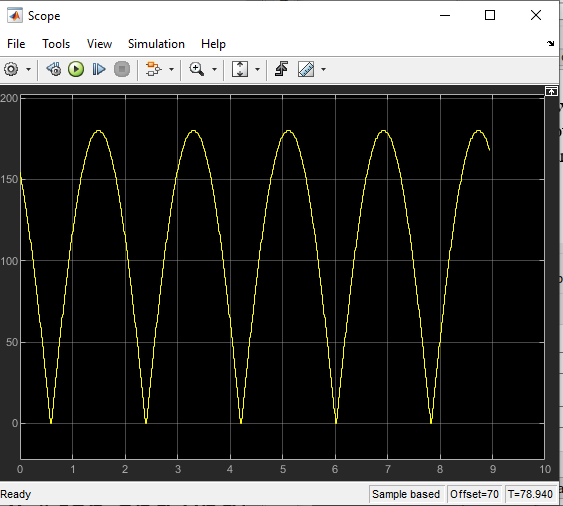
The servo write block takes values ranging from 0 to 180.



If we want the servo rotate following a sinusoidal function . What vectors of output values should be put? attach a screenshot of the ‘desired shaft angle’ block and scope block. Can you observe the sinusoidal rotation?

Answer:





**Exercise 3: code generation**

Go to the folder path:

C:\Users\username\OneDrive\Documents\MATLAB\AtmelAVR\arduino\_servocontrol\_sweep

Find out C-file named ***arduino\_servocontrol\_sweep***

***Copy and Paste the C-code here.***

*/\**

*\* arduino\_servocontrol\_sweep.c*

*\**

*\* Academic License - for use in teaching, academic research, and meeting*

*\* course requirements at degree granting institutions only. Not for*

*\* government, commercial, or other organizational use.*

*\**

*\* Code generation for model "arduino\_servocontrol\_sweep".*

*\**

*\* Model version : 3.5*



*\* Simulink Coder version : 9.4 (R2020b) 29-Jul-2020*

*\* C source code generated on : Mon Mar 1 19:44:59 2021*

*\**

*\* Target selection: ert.tlc*

*\* Embedded hardware selection: Atmel->AVR*

*\* Code generation objectives: Unspecified*

*\* Validation result: Not run*

*\*/*

*#include "arduino\_servocontrol\_sweep.h"*

*#include "arduino\_servocontrol\_sweep\_private.h"*

*/\* Block states (default storage) \*/*

*DW\_arduino\_servocontrol\_sweep\_T arduino\_servocontrol\_sweep\_DW;*

*/\* Real-time model \*/*

*static RT\_MODEL\_arduino\_servocontrol\_T arduino\_servocontrol\_sweep\_M\_;*

*RT\_MODEL\_arduino\_servocontrol\_T \*const arduino\_servocontrol\_sweep\_M =*

*&arduino\_servocontrol\_sweep\_M\_;*

*static void rate\_monotonic\_scheduler(void);*

*/\**

*\* Set which subrates need to run this base step (base rate always runs).*

*\* This function must be called prior to calling the model step function*

*\* in order to "remember" which rates need to run this base step. The*

*\* buffering of events allows for overlapping preemption.*

*\*/*

*void arduino\_servocontrol\_sweep\_SetEventsForThisBaseStep(boolean\_T \*eventFlags)*

*{*

*/\* Task runs when its counter is zero, computed via rtmStepTask macro \*/*

*eventFlags[1] = ((boolean\_T)rtmStepTask(arduino\_servocontrol\_sweep\_M, 1));*

*}*

*/\**

*\* This function updates active task flag for each subrate*

*\* and rate transition flags for tasks that exchange data.*

*\* The function assumes rate-monotonic multitasking scheduler.*

*\* The function must be called at model base rate so that*

*\* the generated code self-manages all its subrates and rate*

*\* transition flags.*

*\*/*

*static void rate\_monotonic\_scheduler(void)*

*{*

*/\* Compute which subrates run during the next base time step. Subrates*

*\* are an integer multiple of the base rate counter. Therefore, the subtask*

*\* counter is reset when it reaches its limit (zero means run).*

*\*/*

*(arduino\_servocontrol\_sweep\_M->Timing.TaskCounters.TID[1])++;*

*if ((arduino\_servocontrol\_sweep\_M->Timing.TaskCounters.TID[1]) > 9) {/\* Sample time: [0.01s, 0.0s] \*/*

*arduino\_servocontrol\_sweep\_M->Timing.TaskCounters.TID[1] = 0;*

*}*

*}*

*/\* Model step function for TID0 \*/*

*void arduino\_servocontrol\_sweep\_step0(void) /\* Sample time: [0.001s, 0.0s] \*/*

*{*

*{ /\* Sample time: [0.001s, 0.0s] \*/*

*rate\_monotonic\_scheduler();*

*}*

*}*

*/\* Model step function for TID1 \*/*

*void arduino\_servocontrol\_sweep\_step1(void) /\* Sample time: [0.01s, 0.0s] \*/*

*{*

*uint8\_T rtb\_FixPtSum1;*

*/\* MultiPortSwitch: '<S1>/Output' incorporates:*

*\* Constant: '<S1>/Vector'*

*\* UnitDelay: '<S2>/Output'*

*\*/*

*rtb\_FixPtSum1 =*

*arduino\_servocontrol\_sweep\_P.DesiredShaftAngle\_OutValues[arduino\_servocontrol\_sweep\_DW.Output\_DSTATE];*

*/\* MATLABSystem: '<Root>/Standard Servo Write' incorporates:*

*\* Constant: '<S1>/Vector'*

*\* MultiPortSwitch: '<S1>/Output'*

*\*/*

*if (rtb\_FixPtSum1 > 180) {*

*rtb\_FixPtSum1 = 180U;*

*}*

*MW\_servoWrite(0, rtb\_FixPtSum1);*

*/\* End of MATLABSystem: '<Root>/Standard Servo Write' \*/*

*/\* Sum: '<S3>/FixPt Sum1' incorporates:*

*\* Constant: '<S3>/FixPt Constant'*

*\* UnitDelay: '<S2>/Output'*

*\*/*

*rtb\_FixPtSum1 = (uint8\_T)((uint16\_T)*

*arduino\_servocontrol\_sweep\_DW.Output\_DSTATE +*

*arduino\_servocontrol\_sweep\_P.FixPtConstant\_Value);*

*/\* Switch: '<S4>/FixPt Switch' \*/*

*if (rtb\_FixPtSum1 > arduino\_servocontrol\_sweep\_P.LimitedCounter\_uplimit) {*

*/\* Update for UnitDelay: '<S2>/Output' incorporates:*

*\* Constant: '<S4>/Constant'*

*\*/*

*arduino\_servocontrol\_sweep\_DW.Output\_DSTATE =*

*arduino\_servocontrol\_sweep\_P.Constant\_Value;*

*} else {*

*/\* Update for UnitDelay: '<S2>/Output' \*/*

*arduino\_servocontrol\_sweep\_DW.Output\_DSTATE = rtb\_FixPtSum1;*

*}*

*/\* End of Switch: '<S4>/FixPt Switch' \*/*

*}*

*/\* Model initialize function \*/*

*void arduino\_servocontrol\_sweep\_initialize(void)*

*{*

*/\* Registration code \*/*

*/\* initialize real-time model \*/*

*(void) memset((void \*)arduino\_servocontrol\_sweep\_M, 0,*

*sizeof(RT\_MODEL\_arduino\_servocontrol\_T));*

*/\* states (dwork) \*/*

*(void) memset((void \*)&arduino\_servocontrol\_sweep\_DW, 0,*

*sizeof(DW\_arduino\_servocontrol\_sweep\_T));*

*/\* Start for MATLABSystem: '<Root>/Standard Servo Write' \*/*

*MW\_servoAttach(0, 4);*

*/\* InitializeConditions for UnitDelay: '<S2>/Output' \*/*

*arduino\_servocontrol\_sweep\_DW.Output\_DSTATE =*

*arduino\_servocontrol\_sweep\_P.Output\_InitialCondition;*

*}*

*/\* Model terminate function \*/*

*void arduino\_servocontrol\_sweep\_terminate(void)*

*{*

*/\* (no terminate code required) \*/*

*}*