

Due Date: 0800 on 01-May-2019 (along with Part II)**Project II – Part I: Networked Control System and Cyber Attack Threats Simulation****OBJECTIVES**

- To identify and understand the hardware necessary to control and sense within a simple cyber-physical system, in this case, a robotic arm
- To simulate a Networked Control System (NCS) using MATLAB and Simulink
- To design PI controllers
- To simulate the launch of cyber attacks to NCSs
- To explore via simulation the effects of external disturbances and different cyber attack models on the closed-loop performance of the system

INTRODUCTION

In this course, you have learned about NCSs including Industrial Control Systems and SCADA Systems. In brief, a NCS is a time-critical and safety-critical control process where the control feedback loop is closed via a real-time communication network. It is comprised of multiple spatially distributed nodes (e.g., sensors, actuators, computers, and controllers) that can send and receive information (e.g., commands and measurement signals) through a shared communication network. The sharing of a common communication networks among different control system components increases the chances of malicious agents tampering with the control system via cyber attacks. The goal of the cyber attacker is to either disrupt the performance of the system, lead the system to failure, or gain control of the physical system.

In this lab exercise, you will simulate the robotic arm system in Figure 1 using MATLAB and Simulink. The system is comprised of a robotic arm (a cylindric-shaped link) mounted on a aluminum frame, a BOSCH BNO055 Inertia Measurement Unit (IMU) Sensor, a Hitech HS-475HB Servo motor, and a set of four mbed microcontrollers mounted on CAN Bus-ready protoboards.

The Bosch BNO055 IMU sensor is an intelligent 9-Axis absolute orientation sensor that integrates a triaxial 14-bit accelerometer, a triaxial 16-bit gyroscope with a range of ± 2000 degrees per second, a triaxial geomagnetic sensor, and a 32-bit ARM Cortex M0+ microcontroller running Bosch Sensortec sensor fusion software, in a single package. It provides angular position information about the arm, in this case pitch. The BNO055 sensor calibrates itself when first powered and provides angular position in radians using the I2C communication protocol.

The Hitech HS-475HB Servo motor represents the actuator of the system. It has a range of rotation of about 180° and accepts a Pulse

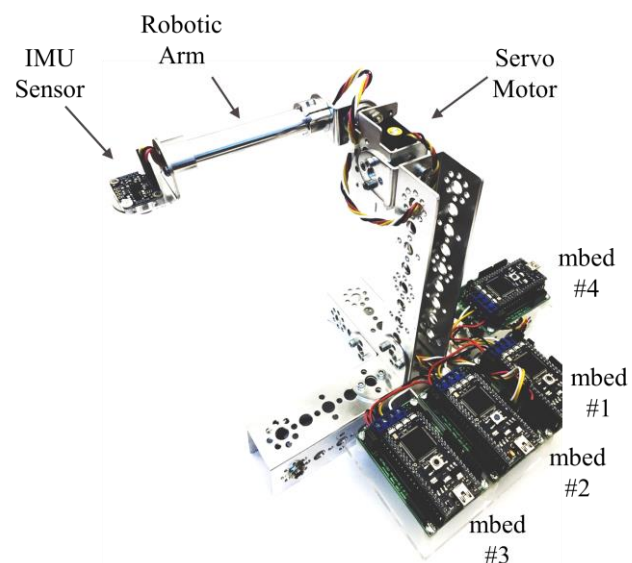


Figure 1. Networked Robotic Arm System

Code Modulated (PCM) signal with a minimum of 900 μs and a maximum of 2100 μs of width, where 900 μs corresponds to positioning the servo motor at 90° and 2100 μs at -90°.

In total, four mbed microprocessors are used, all interconnected using a CAN Bus network. Each mbed has a different function:

- Mbed #1 regulates the position of the servo motor (i.e., actuator) by reading control messages in the **CAN Bus** sent by the controller using pins p29 and p30 and passing a **PCM signal** to the servo (using pin p15)¹ according to the message received.
- Mbed #2 reads the position data (in radians) from the sensor using **I2C** protocol via pins p9 and p10. It then writes a **CAN Bus** message with the sensor data that all other mbeds can read.
- Mbed #3 acts as the controller and is the device implementing the Proportional-Integral control. It reads **CAN Bus** messages with the sensor's ID and compares the measured angular position with the desired one. Based on the error, it determines the control action to take and sends a control message via the **CAN Bus** network.
- Mbed #4 emulates any other device connected to the same **CAN Bus** network, potentially another sensor or actuator controlling a different process. In this lab exercise, mbed #4 will inject false data into the system by posing as the IMU Sensor with the aim of disrupting the closed-loop performance of the system. It can implement two types of attacks:
 - **Stealth Attack:** mbed #4 injects corrupted data into the system by adding a constant bias or measurement error to the sensor data. This attacks slightly corrupts the sensor's reading. Therefore, its effect in the system response is more slowly but harder to identify or detect by conventional fault and false data detectors. Hence the name of stealth attack.
 - **Replay Attack:** mbed #4 starts sending a position (sensor) value from a previously recorded sensor message at a different point in time. This attack effectively breaks the closed-loop system by injecting completely false information. The effects of this attack are more drastic but also easier to detect by conventional fault and false data detectors.

All mbeds are connected to the CAN Bus network using pins p29 and p30. When using a CAN Bus network, all nodes (mbeds) can read and write messages into the bus. Messages from a particular node will carry an ID number to identify the origin. In this lab, mbed #4 will inject data posing as mbed #2.

A functional block diagram of the system is presented in Figure 2, where the external disturbances block represents external unwanted forces on the robotic arm. Similarly, a wiring schematic is given in Figure 3.

EXERCISE

For the purpose of implementing control algorithms for any cyber-physical system and testing its performance under different scenarios, it is extremely useful to develop a mathematical computer simulation model that *approximates* the actual behavior of the system. Simulink is a powerful simulation tool whose programming structure follows similarly to functional block diagrams. In this exercise you will create and simulate operation of the robotic arm system under different scenarios.

¹ Note that mbed's pin p15 is not a regular PwmOut output. Herein, we are assuming we are using the ServoOut.h library, which allows you to use any DigitalOut pin as a PwmOut output by implementing tickers. More details on this will come in Part II of this project.

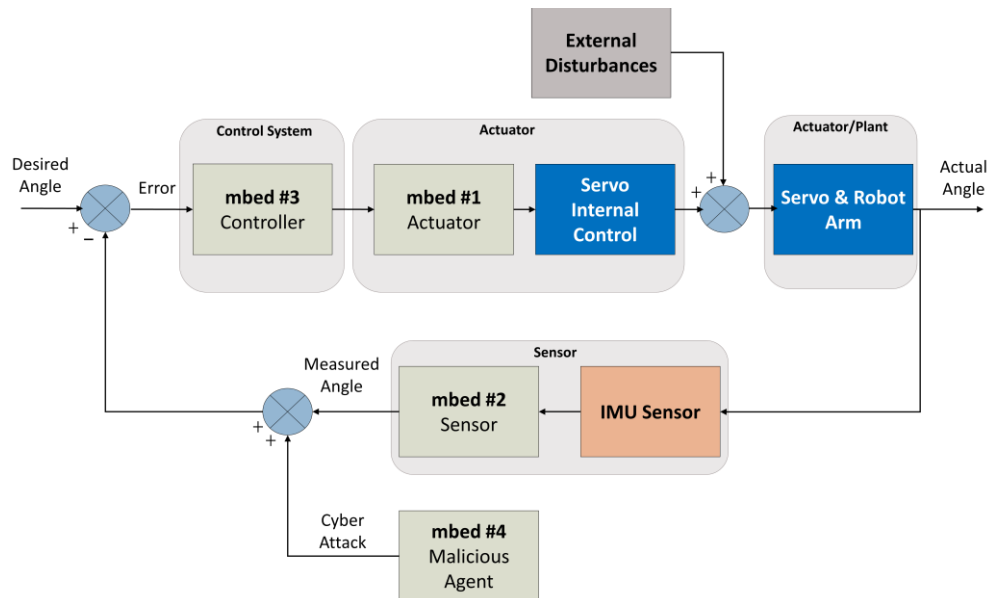


Figure 2. FBD for the NCS

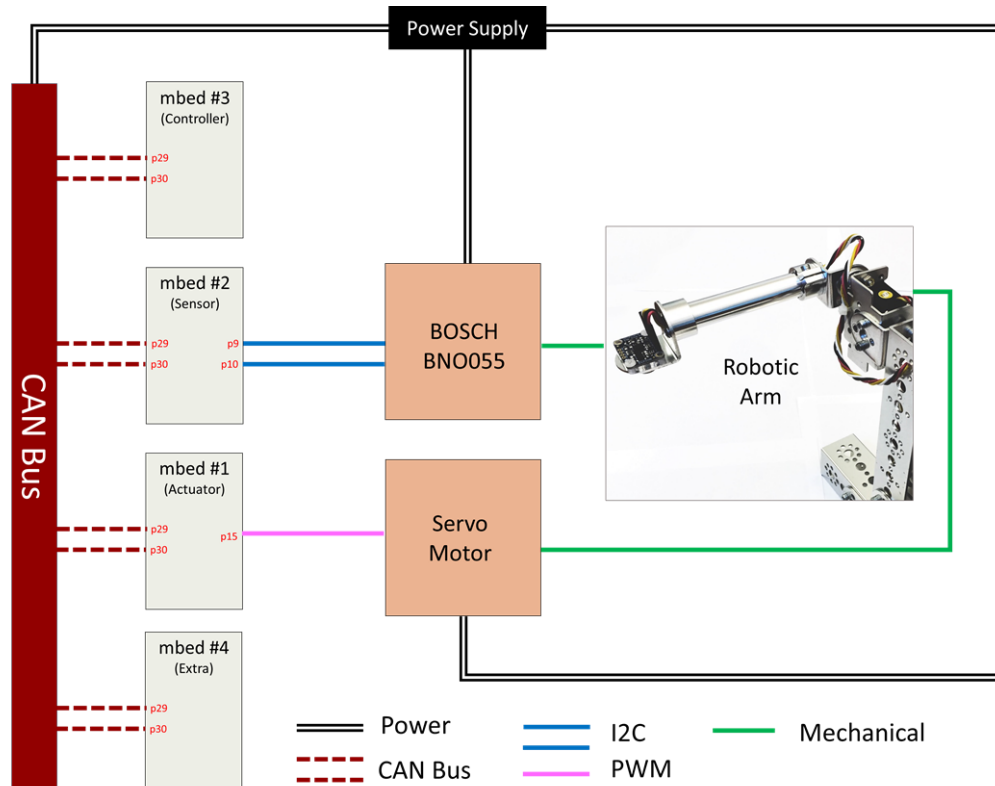


Figure 3. Wiring Schematic

PROCEDURE

Task 1: Creating the Networked Robotic Arm System Simulink Model

1. Create a working directory folder in your laptop to use, e.g., “Project II Part I”.
2. Open the course Google Drive Folder. Go to Projects → Project II → Part I – Simulation and Download the contents of the “Mids Files” folder (6 files in total) to your working directory folder:

- a. SY202_NCS_library.slx
 - b. mbed.jpeg
 - c. bno055.jpg
 - d. servo.jpg
 - e. hand.jpg
 - f. SY202Project.jpg
3. Open MATLAB and specify the directory to your Project II - Part I Files folder location. In order for this lab to work, your **MATLAB working directory should be your newly created folder with the six provided files placed inside the folder.**
4. Open SY202_NCS_library.slx. The library contains blocks that represent the components comprising the robotic arm setup. Each block has specified inputs and outputs.
5. Open a new blank Simulink model template and rename it as ProjPartI_model.slx.
6. Using Figure 4 as a reference, re-create the graphical representation of the system **inside** ProjPartI_model.slx. Drag (or copy/paste) the blocks from SY202_NCS_library.slx.

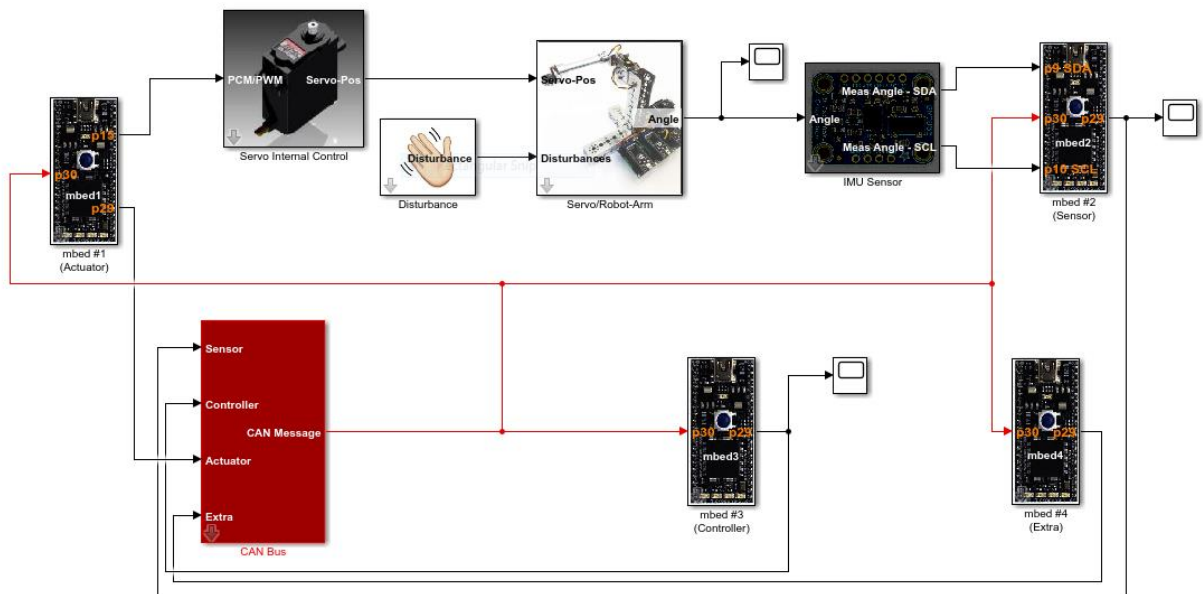



Figure 4. Simulink Model

If you click on each block (mbeds 1, 2, 3, and 4 as well as the disturbance, the servo internal control, the IMU sensor and the robot-arm systems) you will be prompted with different messages and the ability to change some parameters. These parameters can be changed to better reflect the real dynamics of the system. For the moment, keep the default values.

7. To help you visualize the system behavior, we will use scopes and To Workspace blocks (not illustrated in Figure 4). The scope allows you to see the data after running your simulation by double-clicking on it. The To Workspace blocks allow you to save the data for plotting and later analysis. Place one scope and one To Workspace block at each of the following outputs:
 - a. The PCM signal (port: p15) from mbed #1 to the Servo's Internal Control (this signal is the control command sent from the controller to the actuator)
 - b. The disturbance output
 - c. The output (port: angle) of the Servo/Robot-Arm block (this models the actual position of the robotic arm in radians)

- d. The output of mbed #2 (port: p29) (this is the measured angular position of the arm in radians)
8. Double click on the To Workspace blocks and assign an appropriate variable name according to the signal being recorded. Make sure that the “Save format” of all To Workspace blocks is set to “Array”.
9. Show your model to your instructor before proceeding with the next part.

PART 2: Simulation of P and PI control of Robotic Arm System

10. Double-click the Servo/Robot Arm block. Set the initial position to -0.5 radians. Keep all other values at their default.
11. Double-click on mbed #3 (Controller). Set the desired angular position to 0, $K_p = 100$, $K_i = 0$, and the offset to 1400. Simulate the system for at least 40 seconds by clicking on the green button . Double-click on the scope connected to the output (actual angle) of the robotic-arm block. Does the system stabilize at zero? Observe the output of all other scopes.
12. Increase the proportional gain and re-run your simulation. Annotate any observation you make. Is the robotic arm stabilize at 0 radians as desired? What happens to the system as you increase the proportional gain?
13. Set the proportional gain back to 100 and increase the integral gain K_i . Start with $K_i = 20$. Run your simulation for at least 40 seconds and observe the actual position of the robotic arm using the scope.
14. With $K_p=100$, adjust K_i until the system settles near the desired value (e.g., 0 rad) between 5 and 10 seconds. Annotate the value of $K_i=$ _____ and create two plots to be included in your report: one of actual angular position vs time and another from of the control signal (output of mbed #1) vs time. **DO NOT USE the plots from the scopes**, instead use the plot function and the variable saved to your workspace using the To Workspace blocks.
15. Keep the current K_p and K_i values on your mbed #3 controller. Double-click on the IMU sensor. Increase the variance from 0.001 to 0.1. The variance represents the precision of your sensor. A greater variance represents a less precise sensor. Re-run your simulation for different variance and annotate any observations. What happens to the response of the system under steady-state? Once done, reset the variance to 0.001.

PART 3: Simulation of Robotic Arm System with Disturbances

16. Using the values of your PI controller, double-click the disturbance block. Set the start of the disturbance to 20 seconds and the magnitude of the disturbance to 0.5 radians. This scenario represents the sudden motion (rotation by 0.5 radians) of the aluminum frame by someone's action. Observe what happens. Create a plot of actual angular position vs time and another of control signal (output of mbed #1) vs time. Save it for your report.
17. Play different scenarios for the disturbance and observe what happens. At the end, set the start and magnitude of the disturbance back to zero.

PART 4: Simulation of Robotic Arm System under Cyber Attacks

18. Make sure that your mbed #3 controller has been set with your control parameters and that disturbances are set to zero. Double-click on the mbed #4 (Extra) block. You will see different parameters. Rate of attack is the rate of success by the attacker. Setting the rate to zero implies an attacker that remains dormant (i.e., no attack). There are also two types

of attacks to choose and a magnitude for the attack. Simulate the system under different scenarios in Table 1 and fill out the table. **Start all attacks at 20 seconds.** Annotate any observations. Time-to-Steady-State refers to the time at which the actual angular position of the robotic arm reaches “steady-state” minus the start time of the attack (20 seconds). This should be a rough approximation in most cases as the position of the robotic arm may not completely settle. You may need to run the simulation for longer than 40 seconds in order to reach steady-state.

Table 1. Cyber Attack Results

Case	Attack Model	Rate of Attack (%)	Magnitude of Attack (rad)	Actual Angle Steady-State (rad)	Control Signal Steady-State (μ s)	Time-to-Steady-State (s)
1	Stealth	20	0.2			
2	Stealth	20	0.4			
3	Stealth	50	0.2			
4	Stealth	50	0.4			
5	Stealth	80	0.2			
6	Stealth	80	0.4			
7	Replay	20	0.2			
8	Replay	20	0.4			
9	Replay	50	0.2			
10	Replay	50	0.4			
11	Replay	80	0.2			
12	Replay	80	0.4			

19. Create and save plots of angular position vs time and control signal (output of mbed #1) vs time for cases 6 and 12 to be included in your report.

DELIVERABLES

Follow the project report template and the general lab guidelines for SY202 lab reports. Refer to the lab rubric for the grading of the lab report.