

MTRN 3060

PRACTICAL WEEK 7

Vision Sensors

In the previous week, you practiced designing robots and manual control. This week introduces the first step to bringing automation to your robot: the Vision sensor and the workflow to implement the sensor into an automatic robot system.

Learning outcome:

- Students will be able to use Vision sensors in their robotic system.
- Students will practice Object recognition and Line Tracking with Embedded Camera.

TASKS:

Task 1.	Object recognition based on color	3
Task 2.	Line tracking with Pixy Cam	5
Task 3.	Integrate PixyCam in the high-level controller	5

Introduction to Vision sensors

A vision sensor is a device that can capture and process images and output data based on the analysis of the images. A vision sensor can perform various tasks, such as object detection, recognition, measurement, inspection, etc. A vision sensor usually consists of an image sensor, a lens, a processor, and an interface.

An embedded camera is a vision sensor integrated with a computing processor. An embedded camera can perform image processing on the device or send the images to another device for further processing. An embedded camera can have various interfaces, such as USB, SPI, I2C, UART, etc., depending on its compatible platform.

Vision sensors can be used for various applications in various domains, such as industrial, agricultural, medical, smart retail, etc. Some examples are:

1. Quality control systems: Vision sensors and embedded cameras can inspect the quality of products or components in manufacturing or production processes, such as checking for defects, dimensions, colors, etc.
2. Intelligent agriculture and farming: Vision sensors and embedded cameras can monitor and manage crops and livestock such as detecting pests, diseases, weeds, growth stages, etc.
3. Medical and life science: Vision sensors and embedded cameras can perform various types of medical imaging, such as microscopy, multispectral imaging, ophthalmic imaging, dermatology, surgical imaging, etc.
4. Image processing, object recognition, face detection, gesture recognition, depth measurement etc

Pixy cam is a fast vision sensor for DIY robotics and similar applications. It can learn to detect objects you teach by just pressing a button. It can also detect and track lines for use with line-following robots. Pixy cam can connect directly to Arduino or Raspberry Pi using special cables, and it has several interfaces (SPI, I2C, UART, and USB) for communicating with other devices. Pixy cam can simplify your programming by only sending the data you are interested

in, such as the X-Y coordinates, size, and unique ID of the detected objects. Pixy Cam is open source, including its hardware, firmware, and software.

Pixy cam is an example of an embedded camera that can be used for automation. An embedded camera is a camera that is integrated with a computing device or system, such as a microcontroller or a robot.

Prepare:

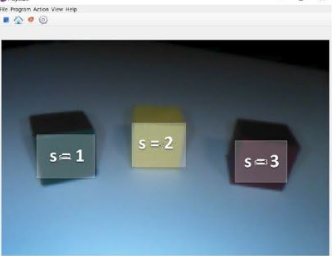
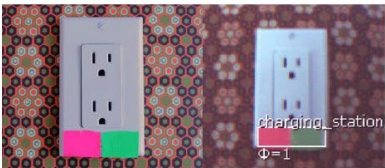
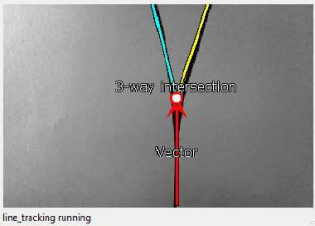
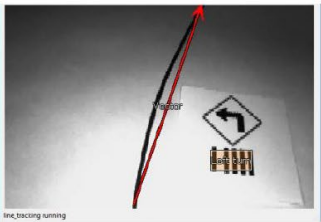
- Install Simulink Support Package for Arduino Hardware
- To see what is on the Camera, you need to install PixyMon:
- <https://pixycam.com/downloads-pixy2/>
- For Arduino code, install the Pixy camera library on Arduino IDE
 - Download the latest Arduino library, "arduino_pixy2-x.y.z.zip" from the link above
 - Bring up the Arduino IDE and import the Pixy library by selecting Sketch→Include Library→Add.ZIP Library... (or if you're using an older version Sketch→Import Library) in the Arduino IDE, and then browse to the Arduino zip file that you just downloaded.

Read through this tutorial: [Get Started with Pixy2 Vision Sensor for Robotics Applications Using Arduino Hardware and Simulink - MATLAB & Simulink - MathWorks Australia.](#)

Introduction to Vision sensor



PIXY CAMERA - BASIC FUNCTION

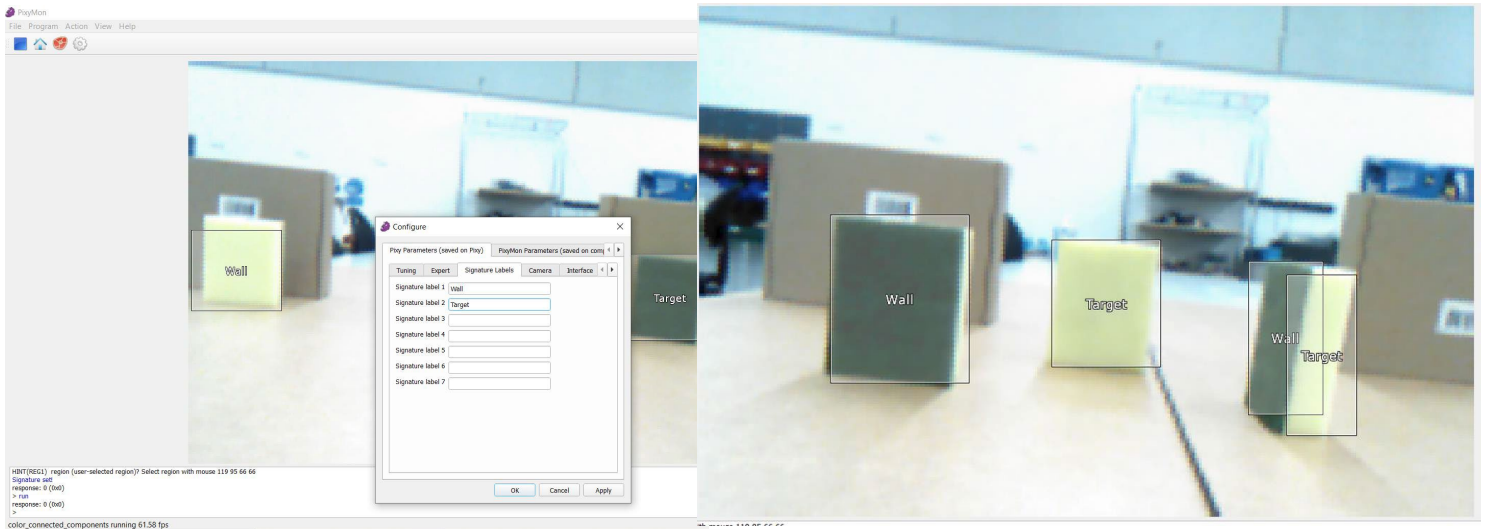
 <p>Object recognition based on Color</p>  <p>Object recognition based on Color Code</p>	 <p>Line tracking running</p> <p>Line tracking</p>  <p>Line tracking and Barcode</p>
--	--

TASK 1. OBJECT RECOGNITION BASED ON COLOR

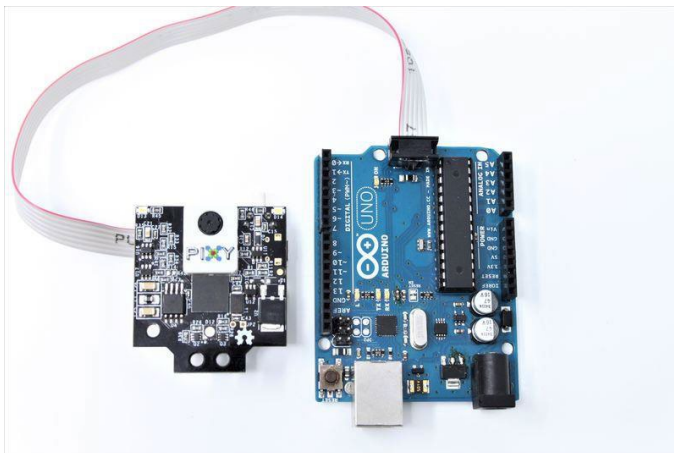
In this task, you will train your Pixy camera, test the embedded Camera through PixyMon, and then use Arduino to determine the sponge's face. Then label the yellow face as the Wall and the green face as the Target.

- 1.1: Connect the Pixy camera and Computer using a Micro USB cable.
- 1.2: Open PixyMonV2 (the image from the Camera should be on the screen).

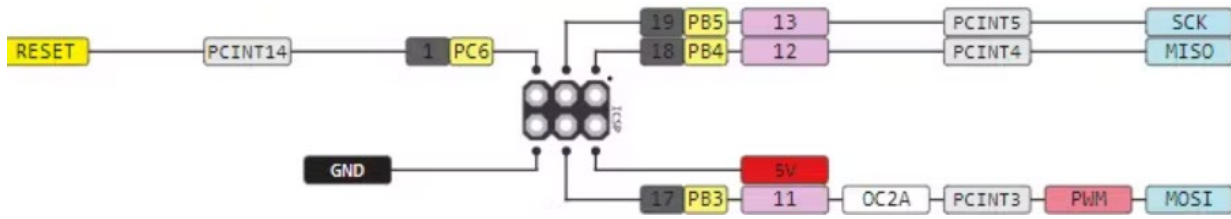
- 1.3: Click on program → object recognize.
- 1.4: Following the instruction to train the Camera to recognize the up face and down face of the sponge:
https://docs.pixycam.com/wiki/doku.php?id=wiki:v2:teach_pixy_an_object_2



- 1.5: Hint: Turn on/off auto white balance and exposure correction might improve the result
- 1.6: After training, download the Pixy library on iLearn and import it to Arduino IDE (Sketch → Include Library → Add. ZIP Library).
- 1.7: The Arduino is connected to the Camera through the ICSP port (ensure cable orientation matches the image below before connecting to the PC to avoid a short circuit), and the communication protocol is SPI.



- 1.8: To read the output data from the Camera, open the ccc_hello_world example code (File→Examples→pixy2).
- 1.9: IMPORTANT: Unplug the camera cable before uploading the program to avoid bricking your Arduino by toggling the RESET pin while uploading.



TASK 2. LINE TRACKING WITH PIXY CAM

- Before starting this task, you need to adjust the camera angle until the line appear on your screen.
- You can try different position and angle to improve the output.



- 2.1: In this task, you will use the line tracking function of Pixy Camera to find the line and read the barcode. The second target is determining the advantages and difficulties between camera mounting points.
- 2.2: Connect the Pixy camera and Computer using a Micro USB cable.
- 2.3: Open PixyMonV2 (the image from the Camera should be on the screen). Click on program → line_tracking
- 2.4: Following the instructions on the website to train the Camera to recognize the line and label the barcode: https://docs.pixycam.com/wiki/doku.php?id=wiki:v2:line_pixymon
- 2.5: To read the output data through Serial Monitor, open the "line_get_all" example code.
- 2.6: (File→Examples→pixy2). The explanation for the function in the code: https://docs.pixycam.com/wiki/doku.php?id=wiki:v2:line_api
- 2.7: To turn on/off the Lamp and LED (try to find the best parameters setting for line tracking): https://docs.pixycam.com/wiki/doku.php?id=wiki:v2:general_api
- 2.8: Example output on Arduino Monitor:

```

COM11
13:36:47.347 -> vector: (5 51) (0 0) index: 0 flags 0
13:36:47.347 -> vector: (6 36) (0 0) index: 0 flags 4
13:36:47.395 -> vector: (5 51) (0 0) index: 0 flags 0
13:36:47.442 -> vector: (6 36) (0 0) index: 0 flags 4
13:36:47.489 -> vector: (5 51) (0 0) index: 0 flags 0
13:36:47.537 -> vector: (5 51) (0 0) index: 0 flags 0
13:36:47.537 -> vector: (5 51) (0 0) index: 0 flags 0
13:36:47.584 -> vector: (6 36) (0 0) index: 0 flags 4
13:36:47.631 -> vector: (5 51) (0 0) index: 0 flags 0
13:36:47.678 -> vector: (5 51) (0 0) index: 0 flags 0
13:36:47.678 -> vector: (6 36) (0 0) index: 0 flags 4
13:36:47.725 -> vector: (5 51) (0 0) index: 0 flags 0
13:36:47.772 -> vector: (5 51) (0 0) index: 0 flags 0
13:36:47.772 -> vector: (6 36) (0 0) index: 0 flags 4
13:36:47.819 -> vector: (5 51) (0 0) index: 0 flags 0
13:36:47.866 -> vector: (5 51) (0 0) index: 0 flags 0
13:36:47.866 -> vector: (5 51) (0 0) index: 0 flags 0
13:36:47.913 -> vector: (5 51) (0 0) index: 0 flags 0
13:36:47.960 -> vector: (5 51) (0 0) index: 0 flags 0
13:36:47.960 -> vector: (5 51) (0 0) index: 0 flags 0
13:36:48.007 -> vector: (5 51) (0 0) index: 0 flags 0
13:36:48.054 -> vector: (5 51) (0 0) index: 0 flags 0
13:36:48.101 -> vector: (5 51) (0 0) index: 0 flags 0
13:36:48.101 -> vector: (5 51) (0 0) index: 0 flags 0
13:36:48.148 -> vector: (5 51) (0 0) index: 0 flags 0
13:36:48.195 -> vector: (6 36) (10 0) index: 0 flags 4
13:36:48.195 -> vector: (5 51) (0 0) index: 0 flags 0
13:36:48.242 -> vector: (6 36) (0 0) index: 0 flags 4
13:36:48.289 -> vector: (6 36) (0 0) index: 0 flags 4
13:36:48.289 -> vector: (5 51) (0 0) index: 0 flags 0
13:36:48.335 -> vector: (6 36) (0 0) index: 0 flags 4
13:36:48.382 -> vector: (6 36) (0 0) index: 0 flags 4
13:36:48.429 -> vector: (5 51) (0 0) index: 0 flags 0
13:36:48.429 -> vector: (6 36) (10 0) index: 0 flags 4
13:36:48.475 -> vector: (6 36) (0 0) index: 0 flags 4
13:36:48.522 -> vector: (6 36) (0 0) index: 0 flags 4
13:36:48.522 -> vector: (5 51) (0 0) index: 0 flags 0
13:36:48.569 -> vector: (5 51) (0 0) index: 0 flags 0
13:36:48.616 -> vector: (5 51) (0 0) index: 0 flags 0
13:36:48.616 -> vector: (5 51) (0 0) index: 0 flags 0
13:36:48.663 -> vector: (5 51) (0 0) index: 0 flags 0
13:36:48.709 -> vector: (5 51) (0 0) index: 0 flags 0
13:36:48.709 -> vector: (5 51) (0 0) index: 0 flags 0

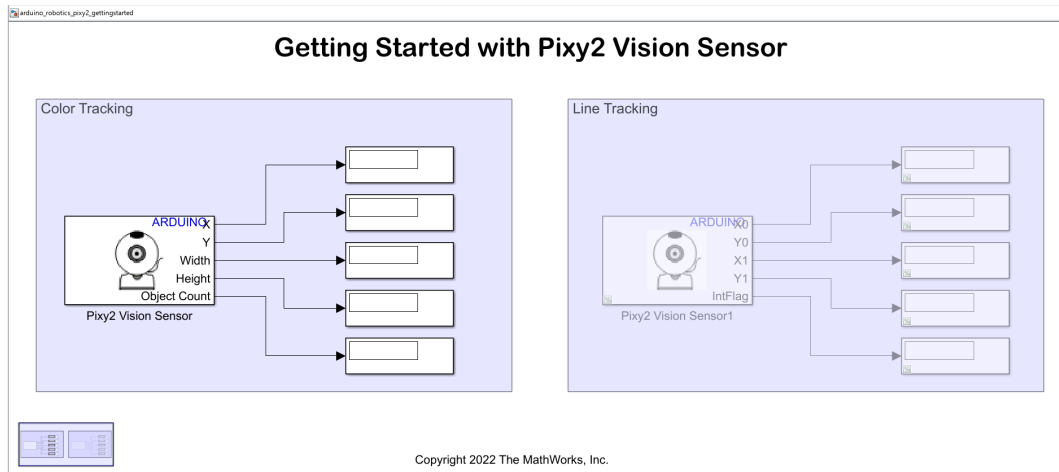
```

TASK 3. INTEGRATE PIXYCAM IN THE HIGH-LEVEL CONTROLLER

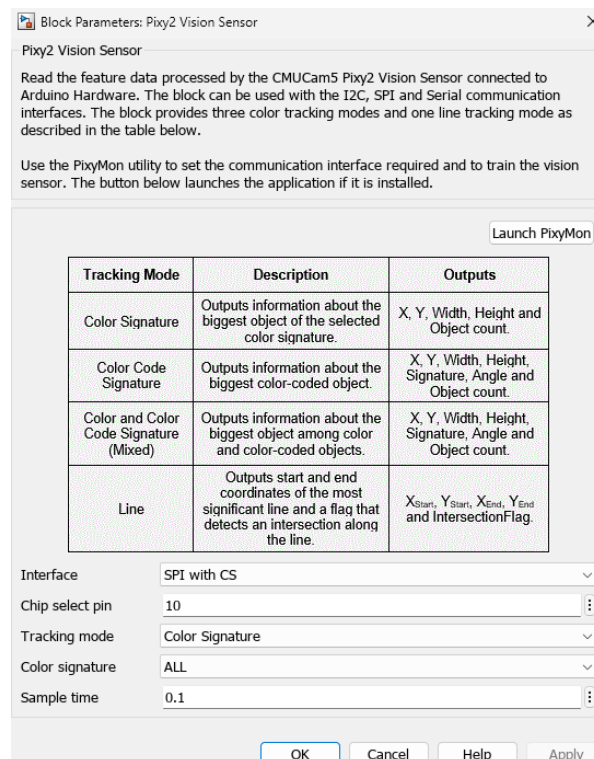
In the real world, different Embedded cameras need additional software support to communicate with the controller. In our case, our controller runs on a Windows computer, and the controller is designed with Matlab, but Matlab doesn't have the support software for communicating with PixyMon. But there is always a workaround solution. We know that Arduino can talk with Pixy Cam, and it also can talk with Matlab. In this exercise, you will practice programming

Arduino so it can act as a low-level controller. You can have low-level controller connect to multiple sensors, process data, and then send it to a high-level controller.

- 3.1: Connect the Pixy2 vision sensor to Arduino using the ribbon cable.
- 3.2: Connect Arduino to PC
- 3.3: Open the Matlab command line and enter:
`openExample('arduino/GetStartedPixy2VisionSensorForArduinoandSimulinkExample')`

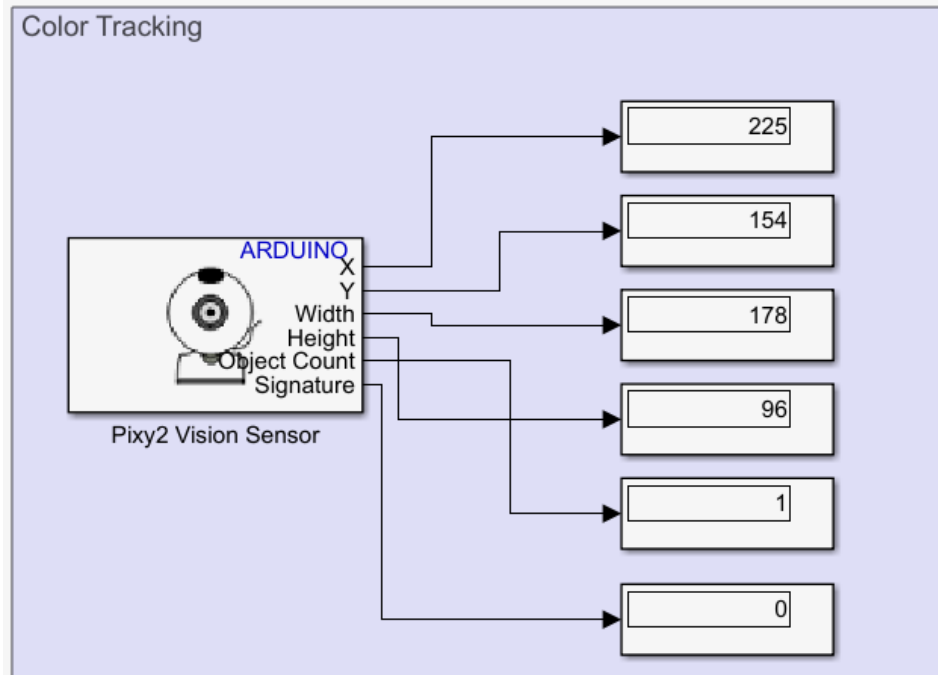


- 3.4: Note: In other Simulink models, Drag and drop the Pixy2 Vision Sensor block from the Simulink library browser to your model.
- 3.5: Configure the block parameters, such as the input port, interface, and tracking mode, to match your Pixy2 vision sensor settings. CS pin should be pin 10 (PixyCam doesn't require a CS pin, but it shouldn't conflict with other pin configurations).



- 3.6: To test the Simulink model → Hardware tab → Monitor and Tune

3.7: The example result:



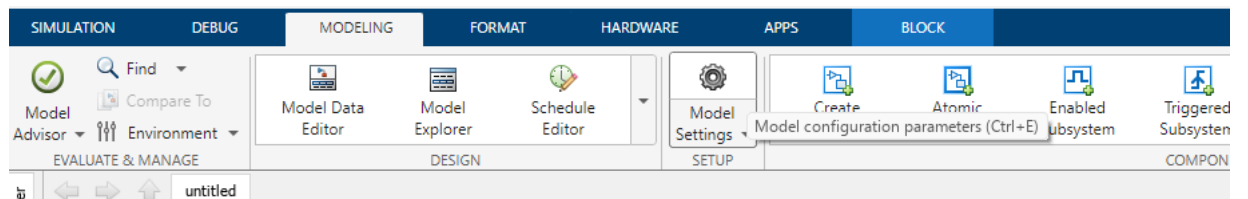
3.8: Mount the Camera on a stationary object.

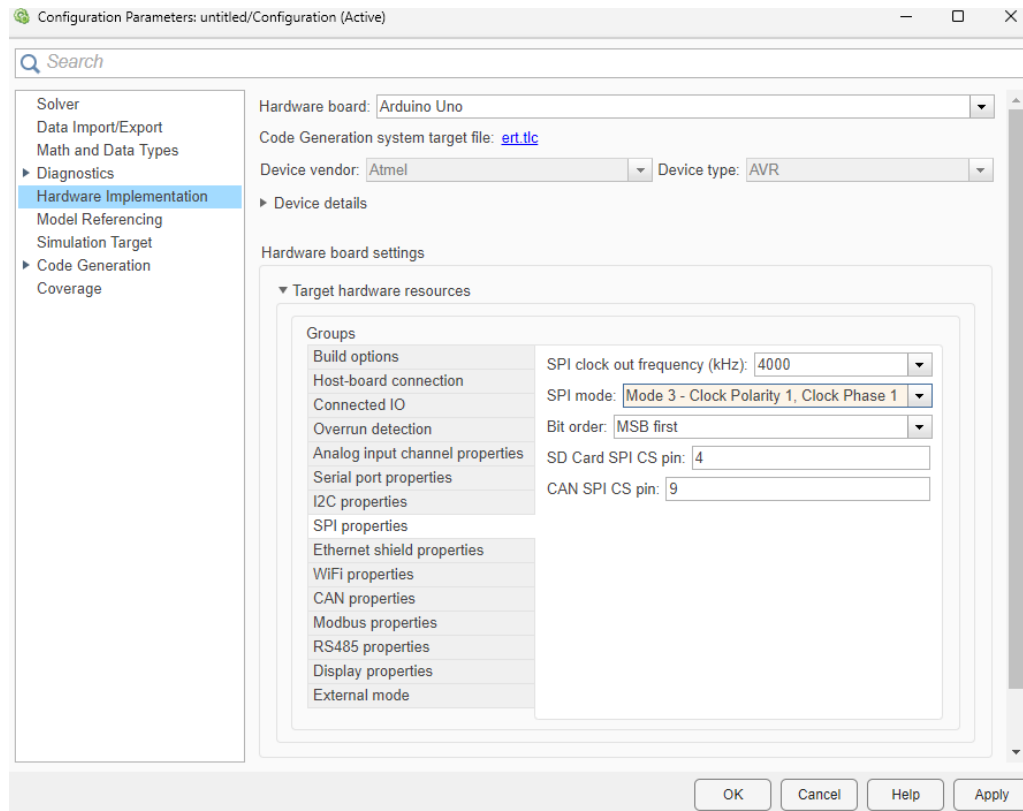
3.9: Copy the Pixy2 Vision Sensor block to other blocks in your Black RobotArm controller model

3.10: For running with external hardware like Arduino, you need to change the hardware implementation to Arduino Uno: Modeling Tab → Model Settings (Ctrl+E) → Hardware implementation:

3.10.1. Hardware board: Arduino Uno

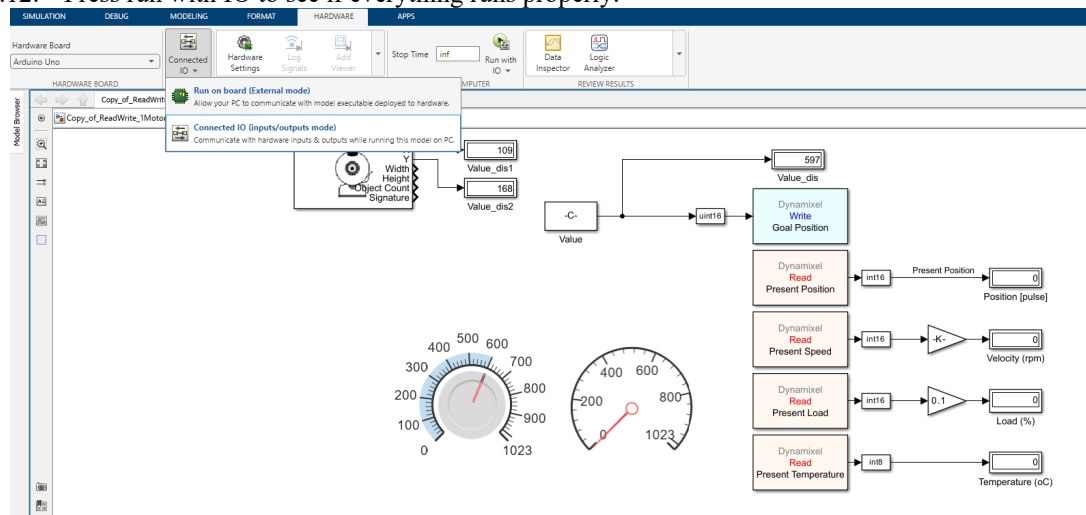
3.10.2. SPI properties (if Camera is connected to SPI bus): SPI mode 3





3.11: Arduino cannot handle the whole Simulink model, and we want to run the model on the Computer and read the value through the IO ports of Arduino, so we switch to Connected IO mode. Go to the Hardware tab → Change Run on board to Connected IO.

3.12: Press run with IO to see if everything runs properly.



3.13: Using the input from the Camera, make the robot point to the yellow Object when detected.

3.14: You can use the output data, such as the coordinates, width, height, and object count of the detected objects, to improve the controller logic.

Challenge: Mount the Camera on top of the robot gripper, then change the controller logic to adapt and make the gripper follow the objective. You will need to implement IK, FK for tracking object translation.