# Arduino

[Arduino Pneumatic (MEGA)](#gjdgxs)

[Arduino Motor (DUE)](#30j0zll)

[Pressure Plate Arduino](#1fob9te)

The Arduinos in the project are equipped with EasyCAT shields, serving as essential intermediaries for communication within the EtherCAT network. Despite their limited processing capabilities, these Arduinos play a crucial role in reading data from various sensors, controlling actuators, and facilitating real-time data exchange. Interfacing with pressure sensors and motors, they contribute to the robot's interaction with its environment. Their primary function includes relaying information between physical components and the MATLAB control system, ensuring synchronization and timely responses. Overall, the Arduinos' integration with EasyCAT shields enhances the system's efficiency and real-time communication capabilities, enabling effective control in a dynamic robotic environment.

# Arduino Pneumatic (MEGA)

1. **Pin Declarations:**
   * Defines pins for valves (A and B) and pressure sensors.
   * Allocates pins for each valve and pressure sensor, ensuring proper interfacing between hardware, Arduino, EtherCat and Matlab.
2. **Pressure Sensor Reading:**
   * Reads analog values from pressure sensors.
   * Converts sensor readings to voltage and updates EasyCAT output buffer.
3. **Valve Control Functions:**
   * Controls solenoid valves based on EasyCAT input buffer.
   * Implements functions for turning specific valves on/off and emptying Pneumatic Artificial Muscles (PAMs).
4. **Setup:**
   * Initializes serial communication at 115200 baud rate.
   * Configures pin modes for valves as OUTPUT.
   * **EtherCAT Initialization:**
     + Utilizes EasyCAT library for LAN9252 communication.
     + Configures EasyCAT with SPI chip select on pin 9.
     + Initializes EasyCAT and enters a loop with blinking LED if initialization fails.
5. **Main Loop:**
   * Executes EasyCAT MainTask to handle communication.
   * **Continuous Operation:** Continuously reads pressure sensor values and controls valves based on EasyCAT communication.

# Arduino Motor (DUE)

1. **Library Inclusions:**
   * Includes necessary libraries: Arduino, MCP\_CAN, MotorHandler, and EasyCAT.
   * Defines VERBOSE\_MODE for optional debugging output.
2. **Definitions and Control Variables:**
   * Sets motor response timeout, communication pins, and motor control parameters.
   * Defines enums for motor CAN IDs, SPI pins, and EtherCAT byte indices.
   * Establishes control variables for motor modes, commands, and responses.
3. **Motor Control Instances:**
   * Initializes communication instances (EasyCAT, MCP\_CAN) and motor control instances (hipMotor, kneeMotor).
   * Configures motor objects with desired parameters.
4. **Communication Functions:**
   * Defines functions to extract XPC commands and check their validity.
   * Implements functions to read commands from EtherCAT and send responses back.
5. **CAN Bus Initialization:**
   * Initializes the CAN bus with a specified baud rate and frequency.
   * Sets the CAN controller to normal mode for message transmission and reception.
6. **Motor Response Handling:**
   * Implements functions to receive and handle motor responses within a specified timeout.
     + Important task to handle 2 motors on the same CAN BUS.
   * Handles response timeouts and prints verbose messages if enabled.
7. **Motor Command Handling:**
   * Defines functions to send motor commands based on the received mode.
   * Handles motor turn-off, position control, torque control, and zero position commands.
8. **Setup Function:**
   * Initializes CAN bus, motor objects, and EtherCAT.
   * Sets up initial motor states and prints setup completion if verbose mode is enabled.
9. **Main Loop:**
   * Executes the EasyCAT MainTask to handle communication.
   * Reads motor commands from EtherCAT, sends commands to motors, and receives motor responses.
   * Sends motor responses back to EtherCAT.
   * Handles verbose mode by incrementing a counter and listening to serial input.

## Pressure Plate Arduino

1. **Force Sensors Configuration:**
   * The Arduino reads data from four force sensors (Force\_sensor\_1 to Force\_sensor\_4) to measure forces applied to the pressure plate.
   * The Arduino is configured to adjust the measurement range using pins A\_z and B\_z.
   * The Operate pin (pin 4) is used to control the operation of the pressure plate.
2. **Emergency Switch:**
   * An emergency switch, connected to pin A1, is monitored, and its analog reading is included in the communication process.
3. **Communication with EasyCAT:**
   * The Arduino communicates with the EasyCAT network, checking for a specific condition (BufferOut.Byte[0] != 1). If this condition is met, it triggers a digital signal on the Operate pin.
   * The force readings from the four sensors are transmitted to the EasyCAT network (BufferIn.Byte[0] to BufferIn.Byte[7]).
   * The analog reading from the emergency switch is also transmitted to the EasyCAT network (BufferIn.Byte[13]).
4. **Initialization:**
   * The Arduino initializes the EasyCAT board, ensuring successful communication. If initialization fails, an error loop with LED blinking is implemented.
5. **Continuous Operation:**
   * The Arduino continuously executes the EasyCAT task and the Communication function in the loop, facilitating real-time communication within the network.