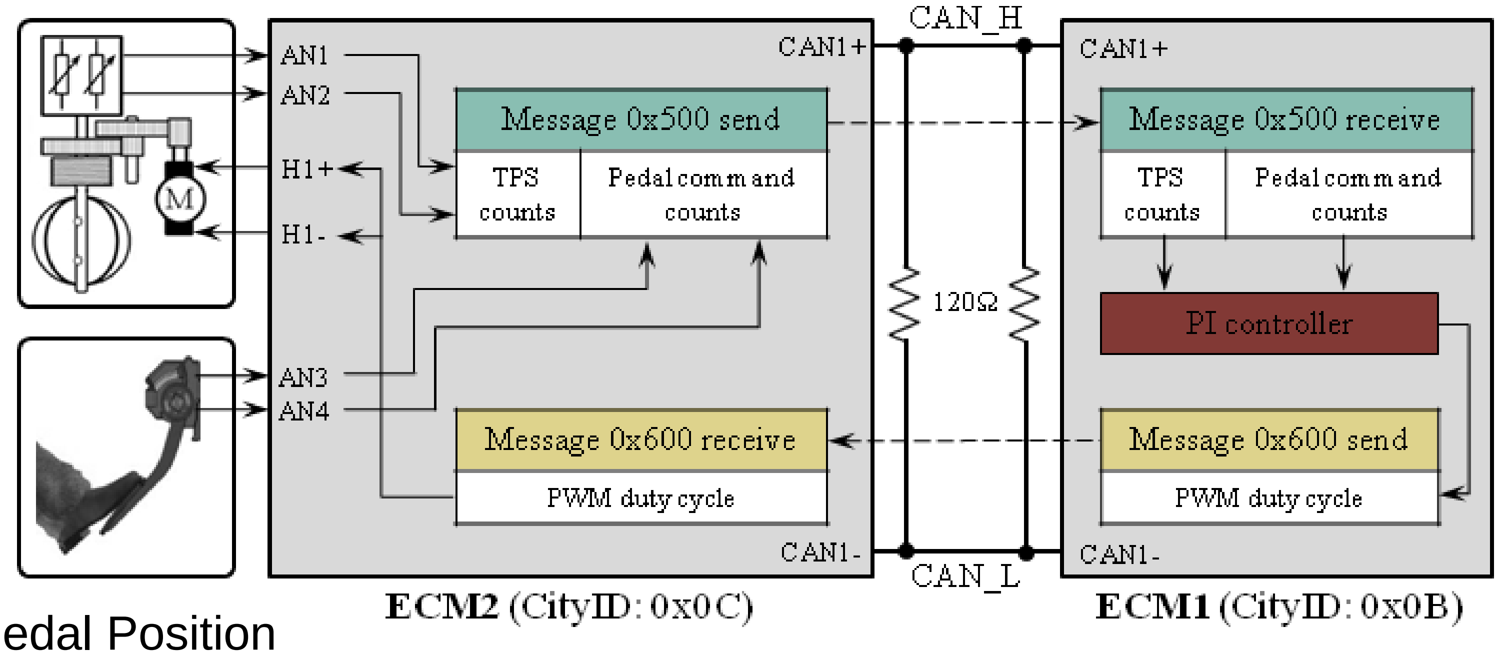
Lab 5: Remote Electronic Throttle Control (ETC) via CAN

Electronic Throttle Body

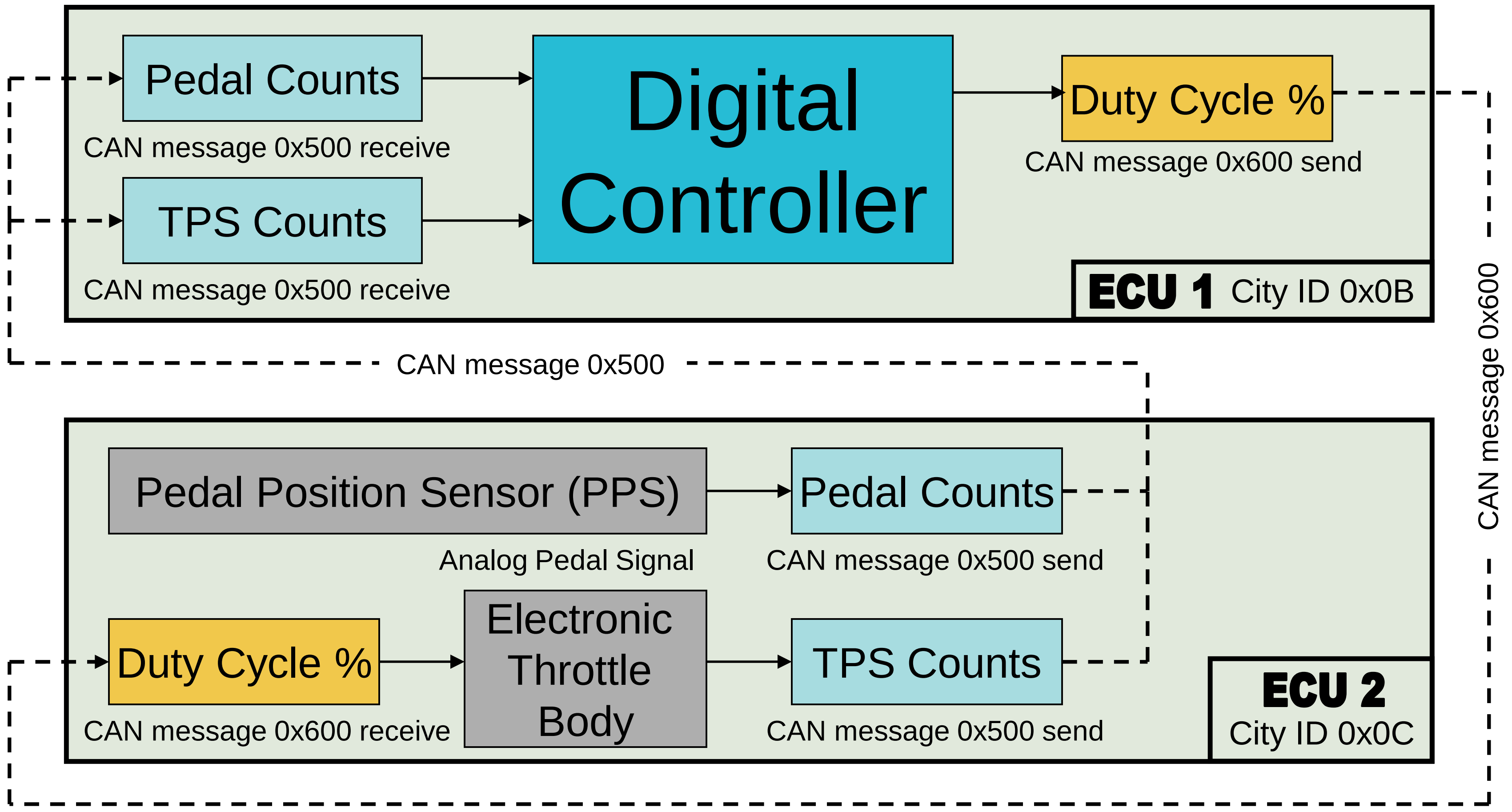


Pedal Position Sensor (PPS)

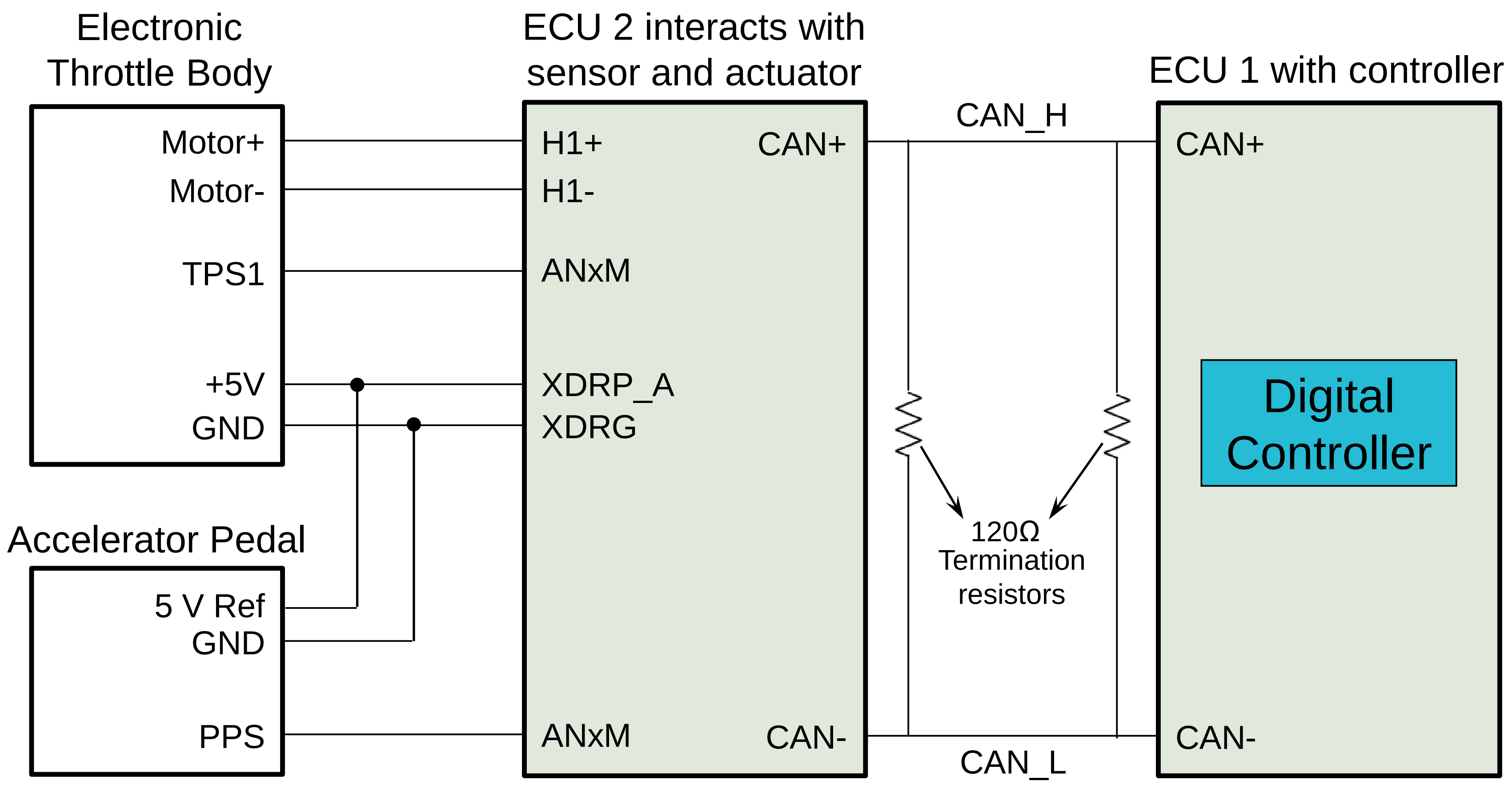
Objectives

1. Develop a remote electronic throttle control system via CAN communication.
2. Learn and practice CAN communication using production Engine Control Units (ECUs).
3. Get familiar with the design procedure of the CAN communication.
4. Gain experience on model development, message definition, endianness, payload manipulation, ID filtering, CAN bus monitoring, and real-time calibration of CAN communication.
5. Compare the pros and cons of the ETC method in lab 4 with remote ETC.

Remote ETC via CAN



Remote ETC Using Mototron

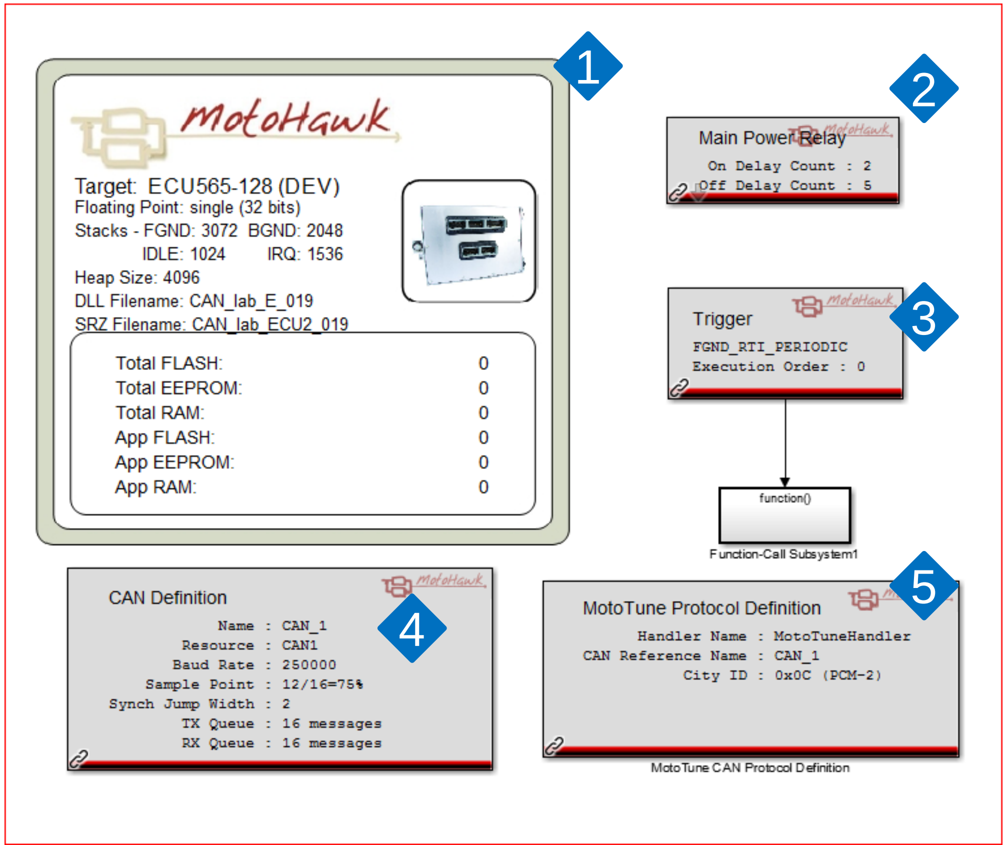


Build Model in ECU 2

1. Use a potentiometer to simulate a pedal position sensor (PPS) for the throttle command. Use a throttle position sensor (TPS) to sense the opening of the throttle plate.
2. Use analog input ports to read in pedal position and throttle plate position.
3. Use an H-bridge to drive the DC servo motor of the throttle body.
4. Use a “MotoHawk Send CAN Message” block to pack the <u>Pedal counts</u> and <u>TPS counts</u> into a CAN message and send the message to ECU 1.
5. Use a “MotoHawk Read CAN Message” block to read the CAN message from ECU 1, which contains the <u>PWM duty cycle</u> of the H-bridge for the throttle servo motor.

Build Model in ECU 2 (Cont.)

6. Add CAN Definition block and MotoTune Protocol Definition block at the top layer of your model as shown below.
7. The rest model refer lab instruction for Lab 4: Electronic Throttle Control. Make sure that the model has a saturation block to limit the duty cycle of the DC motor PWM signals.



```
Target: ECU565-128 (DEV)
Floating Point: single (32 bits)
Stacks - FGND: 3072  BGND: 2048
         IDLE: 1024  IRQ: 1536
Heap Size: 4096
DLL Filename: CAN_lab_E_019
SRZ Filename: CAN_lab_ECU2_019

Total FLASH:      0
Total EEPROM:     0
Total RAM:        0
App FLASH:        0
App EEPROM:       0
App RAM:          0
```

```
Main Power Relay
 On Delay Count  : 2
Off Delay Count  : 5
```

```
Trigger
FGND_RTI_PERIODIC
Execution Order : 0
```

```
CAN Definition
            Name : CAN_1
        Resource : CAN1
       Baud Rate : 250000
    Sample Point : 12/16=75%
Synch Jump Width : 2
        TX Queue : 16 messages
        RX Queue : 16 messages
```

```
MotoTune Protocol Definition
      Handler Name : MotoTuneHandler
CAN Reference Name : CAN_1
           City ID : 0x0C (PCM-2)
```

1. MotoHawk Target Definition block
2. Main Power Relay
3. MotoHawk Function Trigger & Triggered Subsystem (named Foreground)
4. MotoHawk CAN Definition block
5. MotoTune Protocol Definition

Note:
Uses PCM-2
City ID = 0xC

Sample Parameter Setting for CAN Definition Block (ECU 2)

Channel Name	CAN 1
Resource	CAN 1
Bit Timming (Baud Rate)	250Kbaud
Transmit Queue Size	16
Receive Queue Size	16
Install MotoTune Protocol	Yes
City ID	0x0C
City ID Access Level	4

City IDs: The City ID is a MotoTune protocol value that essentially identifies the CAN application modules. Each CAN module must have a unique City ID. In this lab, you may set the City ID of ECU2 as 0x0C.

Build Model in ECU 1

1. Use a “MotoHawk Read CAN Message” block to read and unpack CAN messages from ECU 2. The CAN messages contain pedal and throttle position sensor data.
2. Use a digital controller to determine the duty cycle of the PWM throttle servo control signal based on the difference between the Pedal counts and the TPS counts.
3. Use a “MotoHawk Send CAN Message” block to pack the calculated value of PWM duty cycle into a CAN message and send the message to ECU 2 for the control of the throttle servo motor.
4. Add CAN Definition block and MotoTune Protocol Definition block at the top layer of your model.
5. The rest model refer lab instruction for the Lab 4: Electronic Throttle Control.

Sample Parameter Setting for CAN Definition Block (ECU 1)

Channel Name	CAN 1
Resource	CAN 1
Bit Timming (Baud Rate)	250Kbaud
Transmit Queue Size	16
Receive Queue Size	16
Install MotoTune Protocol	Yes
City ID	0x0B
City ID Access Level	4

Information for Message Definition File

- CAN Message from ECU 2 to ECU 1:
  - ID: 0x500
  - Rate: 10msec
  - Content (Payload Size: 4):
    - Pedal Position, unsigned 10 bits, two bytes from bit 48-bit 63, little endian, ADC counts
    - TPS Position, unsigned 10 bits, two bytes from bit 32-bit 47 , little endian, ADC counts
- CAN Message from ECU 1 to ECU 2:
  - ID: 0x600
  - Rate: 10msec
  - Content (Payload Size: 2):
    - PWM duty cycle of the throttle servo control signal, signed 16 bits, two bytes from bit 48-bit 63, big endian, PWM %

Basic Requirement of Lab and Report

1. Build a CAN network using two ECUs. The ETC controller is implemented in ECU 1 and the ECU 2 is connected to sensors and actuators.
2. Define CAN communication between two ECUs, including Baud Rate, Bus Channel, ID Type, Message ID, Data Length, Repeating Rate, Data Field, and etc. Summarize your definition in a table.
3. Write message definition files in Matlab M-file format. Use two different ways to implement scale and offset for payload manipulation.
4. Develop MotoHawk models for both ECU1and ECU2. Capture these two models and include them in your report. Discuss the details of your model design.

Basic Requirement of Lab and Report

5. Calibrate the upper limit of the PWM duty cycle based on the lab instruction for the Electronic Throttle Control.
6. Calibrate scale, offset, and controller gains to achieve good control performance. Include the calibration results in your report and show the values in the Calibration window.
7. Include charts showing the responses of the throttle position to accelerator pedal command signals. Include corresponding Display windows and Calibration windows in your report.
8. Use CANKing to monitor the CAN bus. Display CAN messages in the output window of the CANKing. Include the output window in your report.

Basic Requirement of Lab and Report

9. Compare CAN data fields in CANKing with the values displayed in MotoTune. Show their relation based on endianness, scale, and offset. Select two values, one Little Endian and one Big Endian.
10. Test the functions of ID filter, age count, and CAN Receive Slot Trigger.
11. Include message definition files for PPS/TPS and PWM Duty Cycle CAN messages in your report.
12. Include one or two additional features of your choice to the project. Indicate these additional features in your report.