MODULE 1

Welcome to Your PEV Brain

Introduction to CAN Bus Communication & Jupyter Notebooks

Lectec PEV AI Curriculum

Day 1 of 14

Complete Slide & Notebook Guide

Integrated Slides + Jupyter Notebook Activities

Table of Contents

Part 1: Slide Content Specifications

This section provides exact text, bullets, headings, and visual descriptions for each slide. Green callout boxes indicate when students should switch to their Jupyter notebooks.

SLIDE 1: Title Slide

Visual Design

- Full-slide gradient background: Dark Blue (#1976D2) to Lectec Blue (#2196F3)

- Yellow accent bar at top (8px height)
- Centered content layout

Text Content

Small Text (Top Center)

MODULE 1

Main Title (Large, Centered)

Welcome to Your PEV Brain

Subtitle

Introduction to CAN Bus Communication

Decorative Element

Yellow horizontal line (120px wide, 4px tall) centered below subtitle

Bottom Text

Lectec PEV AI Curriculum

Day 1 of 14 (bottom right corner)

Speaker Notes

- Welcome students and introduce yourself
- Set expectations: This is a hands-on coding course
- Mention: By the end of 14 days, they will have built a real safety system
- Key phrase: 'Your PEV is about to get a brain - and you're going to program it!'

SLIDE 2: The Hook - What If Your PEV Could Think?

Visual Design

- White background with blue header bar
- Two-column layout: Left = image placeholder, Right = text content
- Yellow callout box at bottom

Header Text

What If Your PEV Could Think?

Body Content

Opening Statement

Imagine riding down the street and your PEV automatically:

Bullet Points (with icons)

- Spots a pedestrian stepping into your path
- Warns you with a beep before you even see them
- Logs the near-miss for your safety review

Yellow Callout Box

By Day 14, YOU will build this system!

Image Placeholder

[Left side: Futuristic PEV with glowing neural network overlay - Nano Banana prompt included in appendix]

Speaker Notes

- This is the 'Start with the End in Mind' moment - show them the exciting destination
- Ask: 'How many of you have almost hit something or been hit while riding?'
- Explain: 'What if your PEV could see danger before you do?'
- Key phrase: 'This isn't science fiction - this is what you're going to build'

SLIDE 3: This Tech Powers Real Vehicles

Visual Design

- White background with blue header bar
- Three equal-width cards in a row
- Third card (Your PEV) highlighted in yellow

Header Text

This Tech Powers Real Vehicles

Card 1: Tesla Autopilot

Background: Light blue (#E3F2FD)

Image: [Tesla interior with Autopilot display]

Title: Tesla Autopilot

Description: Uses CAN bus to read 8 cameras, 12 ultrasonic sensors, and control steering

Card 2: Waymo Robotaxi

Background: Light blue (#E3F2FD)

Image: [Waymo self-driving taxi on street]

Title: Waymo Robotaxi

Description: Fully autonomous vehicles using the same CAN protocol you'll learn today

Card 3: Your Lectec PEV (Highlighted)

Background: Lectec Yellow (#FFCA28)

Image: [Lectec PEV with Raspberry Pi visible]

Title: Your Lectec PEV

Description: Same technology, your hands, real learning - starting today!

Speaker Notes

- Build credibility - this isn't toy technology
- CAN bus was invented in 1986 by Bosch for Mercedes-Benz
- Now used in EVERY modern car, truck, boat, and plane
- Key phrase: 'You're learning industry-standard skills that engineers at Tesla use every day'

SLIDE 4: Today's Mission - Learning Objectives

Visual Design

- White background with blue header bar
- Numbered objectives with circular number badges
- First 3 badges: Blue, Fourth badge: Yellow (highlight)

Header Text

Today's Mission

What you'll accomplish in this module

Objectives List

Objective 1 (Blue badge)

Main text: Explain what CAN bus is

Subtext: Understand the 'nervous system' of your PEV

Objective 2 (Blue badge)

Main text: Identify the three main hardware components

Subtext: Raspberry Pi Zero 2W, VESC Controller, AI Camera

Objective 3 (Blue badge)

Main text: Learn to use Jupyter Notebooks

Subtext: Run code, stop code, and understand cells

Objective 4 (Yellow badge - highlight)

Main text: Read real data from your PEV using Python

Subtext: Voltage, RPM, temperature - live and instant!

Speaker Notes

- These are measurable outcomes - students should be able to do ALL of these by end of class
- Emphasize #4 as the exciting hands-on goal
- Key phrase: 'By the end of today, you will have talked to your PEV and it will have talked back'

SLIDE 5: Your PEV Has a Nervous System

Visual Design

- White background with blue header
- Two-column comparison layout
- Left column: Light blue background (Human Body)
- Right column: Yellow background (Your PEV)
- Large '=' sign between columns

Header Text

Your PEV Has a Nervous System

Left Column: Human Body

Title: Human Body

Image placeholder: [Simple human nervous system diagram]

- Brain = Processes information
- Nerves = Carry signals
- Eyes = See the world
- Muscles = Take action

Center

Large '=' sign in yellow

Right Column: Your PEV

Title: Your PEV

Image placeholder: [PEV system diagram with components labeled]

- Raspberry Pi = The brain

- CAN Bus = The nerves
- AI Camera = The eyes
- VESC Motor = The muscles

Speaker Notes

- This analogy helps students understand the system architecture
- Ask: 'What happens if you cut a nerve in your body?' (Signal can't get through)
- Same thing happens if CAN bus wire is disconnected!
- Key phrase: 'Just like your brain controls your body through nerves, the Pi controls your PEV through CAN bus'

SLIDE 6: Meet Your Hardware Team

Visual Design

- White background with blue header
- Three vertical cards side by side
- Each card has colored header bar, image area, title, and bullet specs

Header Text

Meet Your Hardware Team

Card 1: THE BRAIN

Header bar: Lectec Blue (#2196F3)

Header text: THE BRAIN

Image: [Photo of Raspberry Pi Zero 2W]

Title: Raspberry Pi Zero 2W

Specifications:

- 1GHz quad-core processor
- 512MB RAM
- Runs Python code
- WiFi built-in

Card 2: THE MUSCLES

Header bar: Dark Blue (#1976D2)

Header text: THE MUSCLES

Image: [Photo of VESC motor controller]

Title: VESC Motor Controller

Specifications:

- Controls motor speed
- Reports RPM, voltage, temp
- CAN bus interface
- Real-time telemetry

Card 3: THE EYES

Header bar: Warning Orange (#FF9800)

Card background: Yellow (#FFCA28)

Header text: THE EYES

Image: [Photo of Sony IMX500 AI Camera]

Title: Sony IMX500 AI Camera

Specifications:

- AI built INTO the chip!
- Detects 80 object types
- 30 FPS processing
- Low power consumption

Speaker Notes

- Physical show-and-tell moment - hold up each component if available
- Point out: The AI camera is special because it processes AI ON the chip, not in the cloud
- Ask: 'Why might processing AI on the chip be better than sending to the cloud?' (Speed, privacy, no internet needed)
- Key phrase: 'These three work together as a team - brain, muscles, and eyes'

SLIDE 7: What is CAN Bus?

Visual Design

- White background with blue header
- Two-column layout
- Left: Three stacked info boxes
- Right: Diagram area with light blue background

Header Text

CAN Bus: Controller Area Network

The universal language of vehicles

Left Column: Key Facts

Info Box 1 (Gray background)

Title: Invented in 1986 by Bosch

Subtext: Originally for cars, now everywhere

Info Box 2 (Gray background)

Title: Used in every modern vehicle

Subtext: Cars, trucks, tractors, boats, planes

Info Box 3 (Yellow background - highlight)

Title: Why it matters for YOU

Subtext: Industry-standard skill, real job applications

Right Column: How It Works Diagram

Light blue background box

Title: How CAN Bus Works

[Diagram showing: Two parallel wires with multiple devices connected]

- Two wires: CAN High + CAN Low
- All devices share one 'conversation'
- Messages have IDs for routing

Speaker Notes

- Draw the two-wire bus on the board if possible
- Analogy: 'It's like a group chat where everyone can see all messages, but you only respond to ones with your name'
- The message ID is like having your name mentioned in the group chat
- Key phrase: 'Two wires, infinite possibilities - that's the beauty of CAN bus'

SLIDE 8: How It All Connects

Visual Design

- White background with blue header
- Large diagram area with light gray background

- System architecture showing connections between components

Header Text

How It All Connects

Your PEV's complete nervous system

Diagram Components

Top Row

Blue rounded box: Raspberry Pi Zero 2W

Yellow rounded box: IMX500 AI Camera

Blue line connecting them labeled: CSI Cable

Middle

Large green rounded rectangle spanning width

Label: CAN BUS (2-Wire Network)

Vertical line from Pi down to CAN bus

Bottom

Dark blue rounded box: VESC Motor Controller

Vertical line up to CAN bus

Label below VESC: Speed, RPM, Voltage, Temp

Speaker Notes

- Walk through the data flow: 'When you ask for the battery voltage...'
- Pi sends request over CAN bus → VESC receives → VESC responds → Pi displays
- Camera connects directly to Pi via ribbon cable (CSI), not CAN
- Key phrase: 'This is the nervous system - the Pi is the brain, CAN bus is the nerve, VESC reports the data'

SLIDE 9: Safety First!

Visual Design

- White background with orange header bar (#FF9800)
- Two-column layout with 6 safety rules
- Each rule has a colored left border indicating severity

Header Text

Safety Rules

Essential guidelines for working with your PEV

Safety Rules (2 columns, 3 rules each)

Rule 1: Red border (DANGER)

Title: NEVER ride during coding

Description: PEV must be on a stand or stationary when running code

Rule 2: Orange border (WARNING)

Title: Check battery before starting

Description: Ensure battery is above 20% before lab work

Rule 3: Orange border (WARNING)

Title: Don't touch hot components

Description: Motor and controller get warm during operation

Rule 4: Green border (GOOD PRACTICE)

Title: Always call stop() when done

Description: Clean shutdown protects your hardware

Rule 5: Blue border (INFO)

Title: Ask before motor commands

Description: Reading data is safe, moving motors needs teacher approval

Rule 6: Blue border (INFO)

Title: Report any strange behavior

Description: Weird sounds, smells, or errors? Tell your teacher immediately

Bottom Callout (Orange background)

Safety first = more fun with code later!

Speaker Notes

- Go through each rule and explain WHY it matters
- Rule 1 is most critical - someone could get hurt
- Ask students to repeat back the first rule
- Key phrase: 'The VESC can make the motor spin really fast - we need to be careful'

SLIDE 10: Introduction to Jupyter Notebooks

Visual Design

- White background with blue header

- Screenshot or illustration of Jupyter interface
- Callout boxes explaining interface elements

Header Text

What is a Jupyter Notebook?

Your interactive coding environment

Key Points

A Jupyter Notebook is like a smart document where you can:

- Write and run Python code
- See results immediately below your code
- Add notes and explanations
- Save your work and come back later

Interface Elements to Highlight

- Cells = Individual boxes of code or text
- Run button = Play button that executes code
- Stop button = Square that stops running code
- Output area = Where results appear

Analogy Box

Think of it like a recipe book where you can actually cook each step and see the result!

NOTEBOOK ACTIVITY
Open your Jupyter Notebook now! Your teacher will show you how to access it. Find the file called 'Module_01_PEV_Brain.ipynb'

Speaker Notes

- Have all students open Jupyter before continuing
- Wait until everyone has the notebook open
- Walk around to help anyone having trouble
- Key phrase: 'Jupyter lets you experiment with code - if something doesn't work, just try again!'

SLIDE 11: How to Run a Cell

Visual Design

- White background with blue header

- Step-by-step visual guide with numbered screenshots

Header Text

Running Code in Jupyter

Step-by-Step Instructions

Step 1: Click on a cell

The cell will get a blue or green border when selected

Step 2: Run the cell using ONE of these methods:

- Click the Run button (play triangle) in the toolbar
- Press Shift + Enter on your keyboard
- Press Ctrl + Enter (runs but stays on same cell)

Step 3: Watch for the output

Results appear directly below the cell

The [ ] next to the cell shows [*] while running, then [1], [2], etc. when done

Visual: Show a simple cell before and after running

Before: print('Hello PEV!')

After: Shows 'Hello PEV!' below the cell

NOTEBOOK ACTIVITY
Go to Section A, Cell 1 in your notebook. Click on it and press Shift+Enter. You should see 'Welcome to Jupyter!' appear below.

Speaker Notes

- Demo this live while students follow along
- Walk around and verify everyone sees the output
- Common issue: Students hit Enter instead of Shift+Enter (just makes new line)
- Key phrase: 'Shift+Enter is your new best friend - you'll use it hundreds of times today!'

SLIDE 12: How to Stop Running Code

Visual Design

- White background with orange header (warning context)
- Clear visual of stop button location
- Code example that runs forever

Header Text

Stopping Code That Won't Stop

Sometimes code runs forever - here's how to stop it!

Why Code Might Run Forever

- Loops that never end (while True:)
- Waiting for something that never happens
- Accidentally created infinite loop

How to Stop

Method 1: Click the Stop button (black square) in the toolbar

Method 2: Press the 'I' key twice quickly (I, I)

Method 3: Go to Kernel → Interrupt (menu bar)

Signs Your Code is Still Running

- The cell shows [*] instead of a number
- You can't run other cells
- The circle in the top right is filled (busy)

NOTEBOOK ACTIVITY
Go to Section A, Cell 2 in your notebook. This cell counts forever! Run it, watch it count for a few seconds, then STOP it using the Stop button. Try it!

Speaker Notes

- This is a critical skill - students will need this when they make mistakes
- Let them see the counting for 5-10 seconds before stopping
- Celebrate when they successfully stop it!
- Key phrase: 'The stop button is your emergency brake - always know where it is'

SLIDE 13: What is an API?

Visual Design

- White background with blue header
- Restaurant analogy visual
- Diagram showing: You → Waiter → Kitchen → Waiter → You

Header Text

What is an API?

Application Programming Interface

The Restaurant Analogy

Imagine you're at a restaurant:

- You (the customer) = Your Python code
- The Waiter = The API
- The Kitchen = The VESC motor controller
- The Menu = Available functions you can call

How It Works

1. You look at the menu and tell the waiter what you want
2. The waiter goes to the kitchen and places your order
3. The kitchen prepares your food
4. The waiter brings it back to you

In Code Terms

You don't need to know HOW the kitchen makes the food.

You just need to know WHAT to ask for!

KEY INSIGHT
An API is a 'menu' of commands. You pick what you want, and the API handles all the complicated stuff behind the scenes.

Speaker Notes

- This analogy really resonates with students
- Ask: 'Do you need to know how to cook to order food?' (No!)
- Same with APIs - you don't need to know CAN bus protocols to read voltage
- Key phrase: 'The API is your menu - just pick what data you want!'

SLIDE 14: The VESCStudentAPI

Visual Design

- White background with blue header
- Three category boxes showing API functions

- Color-coded by function type

Header Text

Meet Your API: VESCStudentAPI

Your menu of commands for talking to the VESC

Category 1: Read Functions (Blue box)

Get information FROM the VESC

- get_input_voltage() - Battery voltage
- get_rpm() - Motor speed
- get_motor_current() - Electricity to motor
- get_fet_temperature() - Controller temp
- get_motor_temperature() - Motor temp
- ...and more!

Category 2: Control Functions (Orange box)

Send commands TO the VESC (Teacher approval required!)

- set_duty_cycle() - Control motor speed
- set_current() - Control motor torque
- set_brake_current() - Apply brakes

Category 3: System Functions (Green box)

Manage the connection

- start() - Connect to the VESC
- stop() - Disconnect safely
- is_connected() - Check if working

Speaker Notes

- Today we focus on READ functions - safe and instant
- Control functions come in later modules with safety precautions
- Key phrase: 'Today we're going to READ data - this is completely safe, we're just asking questions'

SLIDE 15: Connecting to Your PEV

Visual Design

- White background with blue header
- Code block showing connection code

- Expected output shown below

Header Text

Let's Connect to the VESC!

The Connection Code

```
from student_api import VESCStudentAPI vesc_api = VESCStudentAPI() vesc_api.start() vesc = vesc_ap
```

Line-by-Line Explanation

- Line 1: Import the API (get the menu ready)
- Line 3: Create our API helper
- Line 4: Start the connection (walk into the restaurant)
- Line 6: Get access to controller #74 (our specific VESC)

Expected Output

```
Connected to VESC controller! Battery: XX.X V
```

If You See an Error

```
VESC controller not found or not responding!
```

This means: Check that the PEV is powered ON and the cable is connected

NOTEBOOK ACTIVITY
Go to Section C in your notebook. Run the connection cell. Raise your hand when you see the green checkmark and battery voltage!

Speaker Notes

- This is a big moment - first time talking to the PEV!
- Walk around and verify everyone gets connected
- Troubleshoot: Power on? Cables connected? Right notebook?
- Key phrase: 'You just said hello to your PEV, and it said hello back with its battery voltage!'

SLIDE 16: Reading Battery Voltage

Visual Design

- White background with blue header
- Code block with explanation
- Voltage interpretation guide

Header Text

Your First Data Read: Battery Voltage

The Code

```
voltage = vesc.get_input_voltage() print(f"Battery Voltage: {voltage} V")
```

What This Does

5. Asks the VESC: 'What's the battery voltage right now?'
6. VESC responds over CAN bus with the number
7. We store it in a variable called 'voltage'
8. We display it using print()

Understanding the Reading

Voltage Interpretation Guide:

- 36V+ = Fully charged (10S battery)
- 33-36V = Good charge remaining
- 30-33V = Getting low, consider charging
- Below 30V = Low battery warning!

NOTEBOOK ACTIVITY
Now it's YOUR turn! In Section D, Cell 1, write your own code to read the voltage. Fill in the blank: voltage = vesc.____________() Then print it out!

Speaker Notes

- Students write their own code here - not just running pre-written code
- The blank is: get_input_voltage
- Walk around and check their work
- Key phrase: 'You just asked your PEV a question and got a real answer - that's programming!'

SLIDE 17: Reading Motor RPM - Part 1

Visual Design

- White background with blue header
- Two states shown: Motor stopped vs Motor spinning

Header Text

Reading Motor Speed (RPM)

RPM = Revolutions Per Minute

The Code

```
rpm = vesc.get_rpm() print(f"Motor RPM: {rpm}")
```

What to Expect

When motor is NOT spinning:

Output: Motor RPM: 0

This is normal! No movement = 0 RPM

When motor IS spinning:

Output: Motor RPM: 2847 (or some other number)

Positive = Forward direction

Negative = Reverse direction

Key Point

REAL-TIME DATA
The RPM reading is INSTANT - it shows what's happening RIGHT NOW.
Every time you run the code, you get the current value.

NOTEBOOK ACTIVITY
In Section D, Cell 2, run the RPM code with the motor NOT spinning.
Write down what you see: _________ RPM

Speaker Notes

- First reading will be 0 - that's expected!
- We'll prove it's real-time in the next slide
- Key phrase: 'Zero RPM makes sense - the motor isn't moving!'

SLIDE 18: Reading Motor RPM - Part 2 (The Proof!)

Visual Design

- White background with green header (activity)
- Before/After comparison

- Hand-spinning illustration

Header Text

Proving It's Real-Time!

Let's see the data change instantly

The Experiment

Step 1: Run the code with motor stopped

Expected result: 0 RPM

Write it down: _________ RPM

Step 2: Gently spin the wheel BY HAND

Keep it spinning slowly and safely

Step 3: While spinning, run the code AGAIN

Expected result: A number OTHER than 0!

Write it down: _________ RPM

What This Proves

The code reads what's happening RIGHT NOW!

This is called REAL-TIME DATA - instant feedback from your PEV.

NOTEBOOK ACTIVITY
Time for the experiment! Run the RPM cell, write down the number. Then GENTLY spin the wheel by hand and run it AGAIN while it's spinning. Write both numbers in your notebook!

Speaker Notes

- This is an 'aha!' moment - data is live!
- Supervise the wheel spinning - gentle and safe
- Students should see different numbers
- Key phrase: 'See how it changed? That's real data from your real PEV, right now!'

SLIDE 19: Fill-in-the-Blank Challenge: Temperature

Visual Design

- White background with blue header
- Code block with blanks
- Hint section

Header Text

Your Turn: Read the Temperature!

The Challenge

Fill in the blanks to read the controller (FET) temperature:

```
temp = vesc.get_______temperature() print(f"Controller Temperature: {_____} °C")
```

Hints

- Hint 1: The FET is the electronic part of the controller
- Hint 2: Look at the API list from earlier - what function reads FET temp?
- Hint 3: The second blank should be the variable name

Answer Check

Your output should look like: Controller Temperature: XX.X °C

Normal reading: 20-50°C when idle

NOTEBOOK ACTIVITY
Go to Section E, Cell 1 in your notebook. Fill in the blanks and run the code. Write down the temperature you get: ________ °C

Speaker Notes

- Give students 2-3 minutes to try
- Answer: get_fet_temperature() and temp
- Walk around and help those who are stuck
- Key phrase: 'You're not just running code anymore - you're WRITING code!'

SLIDE 20: Matching Challenge: Components and Functions

Visual Design

- White background with blue header
- Two-column matching exercise
- Can be done verbally as class or in notebook

Header Text

Match the Component to the Function!

Left Column: What do you want to know?

9. How fast is the motor spinning?
10. How much battery do I have?
11. Is the controller getting hot?
12. How much electricity is the motor using?
13. How hot is the motor itself?

Right Column: Which function to use?

A. get_motor_temperature()

B. get_rpm()

C. get_input_voltage()

D. get_motor_current()

E. get_fet_temperature()

Answers (Teacher Reference)

1-B, 2-C, 3-E, 4-D, 5-A

NOTEBOOK ACTIVITY
Complete the matching exercise in Section F of your notebook. Then run the 'check answers' cell to see if you got them right!

Speaker Notes

- Can do this as a class verbally before notebook
- Call on students to share their answers
- Discuss why each match makes sense
- Key phrase: 'Now you know which question to ask to get the answer you need!'

SLIDE 21: What You Learned Today - Summary

Visual Design

- White background with blue header
- Checklist format with green checkmarks
- Key terms sidebar

Header Text

What You Learned Today

Achievement Checklist

- CAN bus is the 'nervous system' of vehicles
- Three components: Pi (brain), VESC (muscles), Camera (eyes)
- How to use Jupyter Notebooks (run, stop, restart)
- What an API is and why we use one
- Connected to your PEV with Python
- Read real voltage, RPM, and temperature data
- Proved that the data is REAL-TIME

Key Terms Sidebar

CAN Bus

Controller Area Network - the two-wire communication system

VESC

Motor speed controller that reports telemetry

API

Application Programming Interface - your 'menu' of commands

Telemetry

Data sent from the VESC (voltage, RPM, temp, etc.)

Real-Time

Data that reflects what's happening RIGHT NOW

Speaker Notes

- Review each item and ask students to explain
- Celebrate their accomplishments!
- Key phrase: 'You talked to your PEV today, and it talked back. That's real engineering!'

SLIDE 22: Coming Up Next - Module 2

Visual Design

- Blue gradient background (like title slide)
- Preview of next module
- Yellow accent bar at bottom

Small Text (Top)

COMING UP IN MODULE 2

Main Title

Building a Live Dashboard

Subtitle

Real-time data visualization with continuous updates

Preview Boxes

- Continuous monitoring loops
- Updating displays
- Data logging

Bottom Encouragement

Great job today!

You've taken your first step into vehicle communication

Speaker Notes

- Build excitement for the next session
- Mention: Tomorrow we'll make the data update automatically
- Thank students for their focus and participation
- Key phrase: 'Today you asked single questions. Tomorrow, you'll have a continuous conversation!'

Part 2: Jupyter Notebook Specifications

This section provides the complete cell-by-cell content for the Module 1 Jupyter Notebook.

Section A: Jupyter Basics

Purpose: Teach students how to use Jupyter before introducing VESC content

Cell A.1 - Welcome Message (Markdown)

Type: Markdown

Content:

```
# Welcome to Jupyter Notebooks! This is your interactive coding environment. Let's learn how to use it!
```

Cell A.2 - First Code Cell

Type: Code

Purpose: Test that they can run a cell

```
# Your first code cell! # Click on this cell, then press Shift+Enter to run it print("Welcome to Jupyter!") pr
```

Expected output: Welcome to Jupyter! / If you can see this, you ran the cell successfully!

Cell A.3 - Counting Forever (For Stop Practice)

Type: Code

Purpose: Practice stopping code

```
# This code counts forever - you'll need to STOP it! # Run this cell, watch it count, then click the STOP bu
```

Instructions: Run it, let it count to 10 or so, then stop it

Cell A.4 - Simple Math

Type: Code

Purpose: Show that Python can do calculations

```
# Python can do math! apples = 5 oranges = 3 total_fruit = apples + oranges print(f"I have {apples} apple
```

Cell A.5 - Variables Practice

Type: Code

Purpose: Practice creating and using variables

```
# YOUR TURN: Change the values below and run again! my_name = "Student" # Change this to your na
```

Cell A.6 - Check Understanding (Markdown)

Type: Markdown

Content:

```
## Quick Check! **Answer these questions:** 1. What keyboard shortcut runs a cell? ___________________
```

Section B: More Practice (Non-VESC)

Purpose: Build confidence with Python before hardware

Cell B.1 - Functions Introduction

Type: Code

```
# Functions are like recipes - they do something when you call them def say_hello(name): print(f"Hello, {na
```

Cell B.2 - Function with Return Value

Type: Code

```
# Some functions give you a value back (return) def add_numbers(a, b): result = a + b return result # Use
```

Cell B.3 - Practice Creating a Function

Type: Code

```
# YOUR TURN: Fill in the blank to make this function work # The function should multiply two numbers
```

Section C: Connecting to the VESC

Purpose: Establish connection to real hardware

Cell C.1 - Section Header (Markdown)

Type: Markdown

```
#   Connecting to Your PEV Now we'll connect to real hardware! Make sure your PEV is powered ON.
```

Cell C.2 - Import and Connect

Type: Code

```
# Setup and connect to the VESC import sys import os sys.path.append(os.path.dirname(os.path.dirname(o
```

Cell C.3 - Checkpoint (Markdown)

Type: Markdown

```
##   Checkpoint! **Did you see the green checkmark and battery voltage?** - YES → Great! Continue to S
```

Section D: Reading Telemetry Data

Purpose: Students read real data from their PEV

Cell D.1 - Read Voltage (Fill in Blank)

Type: Code

```
# YOUR TURN: Fill in the blank to read the voltage # Hint: Look at what function we used in the connect
```

Answer: get_input_voltage

Cell D.2 - Read RPM

Type: Code

```
# Read the motor RPM (Revolutions Per Minute) rpm = vesc.get_rpm() print(f"Motor RPM: {rpm}") #   I
```

Cell D.3 - RPM Experiment Instructions (Markdown)

Type: Markdown

```
##   RPM Experiment 1. Run the cell above with the motor NOT spinning → Write down the RPM 2. GEN
```

Cell D.4 - Read Motor Current

Type: Code

```
# Read how much electricity the motor is using current = vesc.get_motor_current() print(f"Motor Current:
```

Section E: Fill-in-the-Blank Challenges

Purpose: Students write their own code with guidance

Cell E.1 - Temperature Challenge

Type: Code

```
# CHALLENGE: Read the controller (FET) temperature # Fill in BOTH blanks! temp = vesc.get________te
```

Answers: get_fet_temperature() and temp

Cell E.2 - Motor Temperature Challenge

Type: Code

```
# CHALLENGE: Now read the MOTOR temperature motor_temp = vesc.get____________________() # Wh
```

Answer: get_motor_temperature

Cell E.3 - Multiple Readings Challenge

Type: Code

```
# BOSS CHALLENGE: Read THREE values at once! # Fill in all the blanks voltage = vesc.____________
```

Answers: get_input_voltage, get_rpm, get_fet_temperature

Section F: Matching Exercise

Purpose: Reinforce understanding of which function gets which data

Cell F.1 - Matching Instructions (Markdown)

Type: Markdown

```
## Matching Challenge! Match each question to the correct function. **Questions:** 1. How fast is the mo
```

Cell F.2 - Student Answers

Type: Code

```
# Enter your answers below (just the letters) # Example: answer_1 = ”B” answer_1 = ”____” # How fast
```

Cell F.3 - Check Answers

Type: Code

```
# Check your answers! correct = { 1: ”B”, # RPM 2: ”C”, # Voltage 3: ”E”, # FET temp 4: ”D”, # Moto
```

Section G: Wrap-Up

Cell G.1 - Clean Shutdown

Type: Code

```
# Always clean up when you’re done! vesc_api.stop() print(” VESC connection closed safely”) print(”Great
```

Cell G.2 - Summary (Markdown)

Type: Markdown

```
## Module 1 Complete! ### What you learned: - How to use Jupyter Notebooks - How to run and st
```

Part 3: Teacher Notes & Timing

Suggested Timing (45-minute class)

Opening & Hook (Slides 1-3): 5 minutes

- Get students excited with the 'what if' scenario
- Show real-world applications to build credibility

Objectives & Hardware (Slides 4-8): 8 minutes

- Clear learning objectives
- Hardware show-and-tell if possible
- System architecture overview

Safety Rules (Slide 9): 3 minutes

- Critical - ensure understanding
- Have students repeat Rule 1

Jupyter Basics (Slides 10-12 + Notebook A-B): 10 minutes

- Hands-on practice with run/stop
- Everyone should complete Section A before proceeding

API & Connection (Slides 13-15 + Notebook C): 7 minutes

- API explanation with restaurant analogy
- Everyone should be connected before proceeding

Reading Data (Slides 16-18 + Notebook D): 8 minutes

- Voltage reading
- RPM experiment with hand-spinning

Challenges & Summary (Slides 19-22 + Notebook E-F): 4 minutes

- Fill-in-the-blank exercises
- Matching game
- Celebration and preview

Common Issues & Solutions

Issue: Student can't connect to VESC

- Check: Is the PEV powered ON?
- Check: Is the USB cable connected?
- Check: Is the correct notebook open?
- Try: Restart the kernel and run connection cell again

Issue: Student accidentally closed connection

- Solution: Run the connection cell (C.2) again

Issue: Code won't stop running

- Click Stop button OR press I,I OR Kernel → Interrupt
- If that fails: Kernel → Restart

Issue: Student gets all zeros for readings

- This is NORMAL for RPM and current when motor isn't moving
- Voltage should always show a value - if not, check connection

Key Phrases to Use

- 'Your PEV is about to get a brain - and you're going to program it!'
- 'The API is your menu - just pick what data you want!'
- 'You talked to your PEV today, and it talked back. That's real engineering!'
- 'Shift+Enter is your new best friend'
- 'The stop button is your emergency brake'
- 'Zero RPM makes sense - the motor isn't moving!'