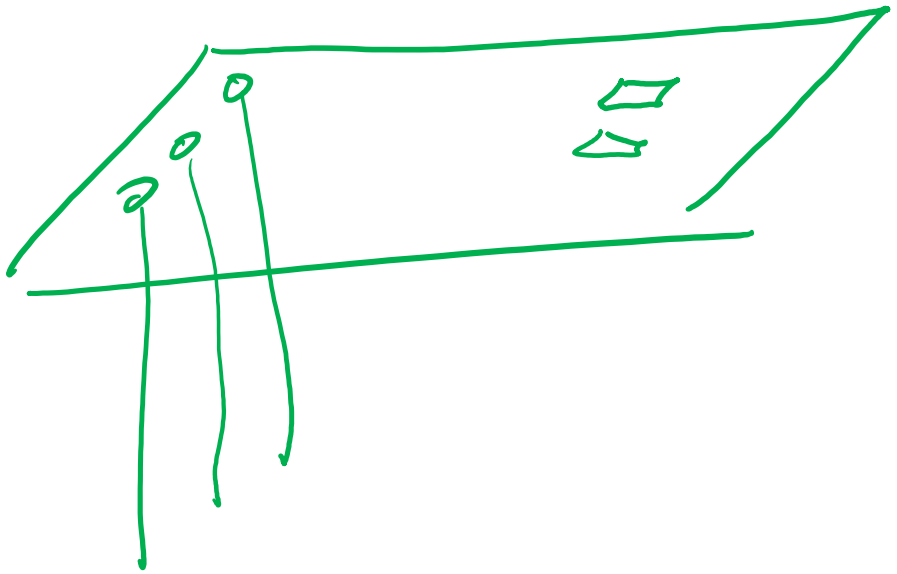
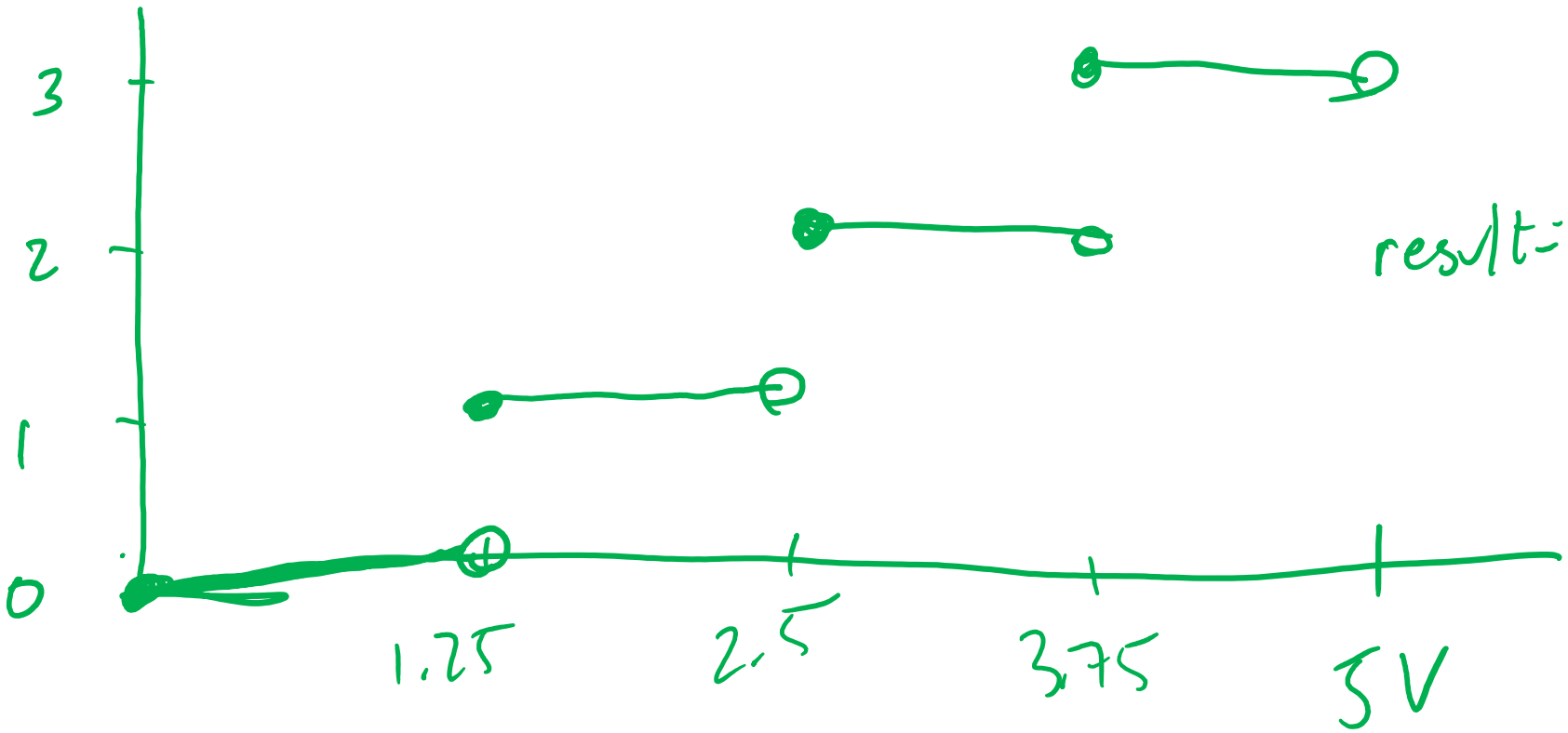
Questions?



2-bit ADC



result = floor($\frac{V}{5V} \cdot 4$)

Integration and Testing

$$\int_0^{7\ \text{weeks}} \text{parts}(\text{VEX}) \times \text{testing}(t) \cdot dw = \text{Success!}$$

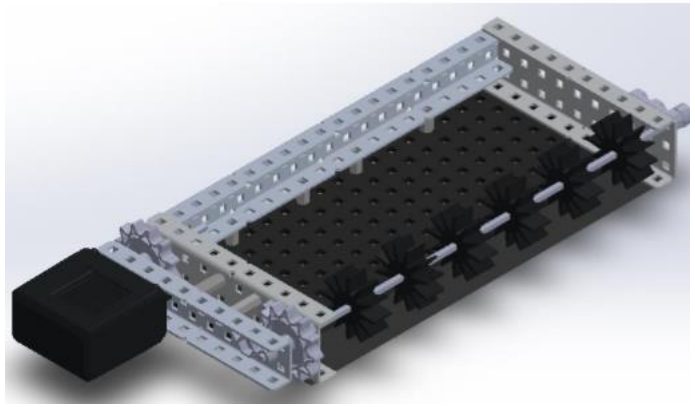
System integration

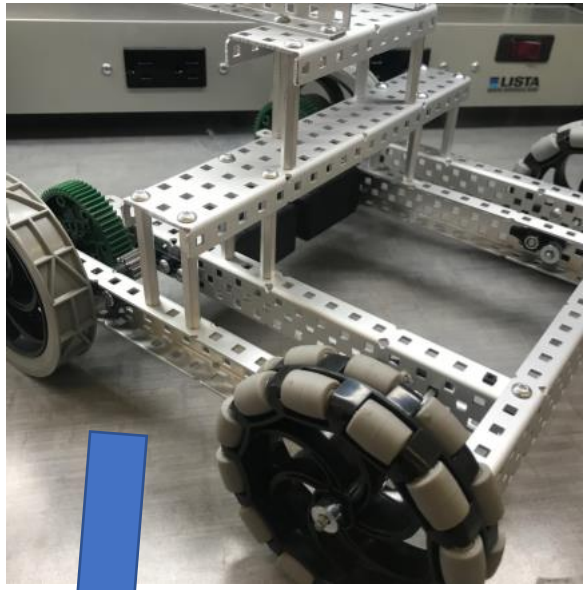
Up til now, you've been focusing your testing on specific subsystems:

- Will the chassis be stable?
- Can a motor provide enough torque?
- How fast is your delivery mechanism?
- How reliable is your pizza holder?
- Etc.

You need to integrate all of the subsystems into a working system.

System integration





?

The focus of your testing shifts in the integration phase

- Subsystems
  - Can this work?
  - What does it take to work well?
  - How does this work best?

- Integrated system
  - Does this *still* work?
  - Are there any unintended interactions?
  - *Can I be confident that the system will perform*?

What does it take to get this to succeed?

What does it take to make this to fail?

IDR and creating test plans for your system

- The IDRs will be in the second half of next week
  - You will demonstrate an integrated physical system by delivering a pizza in teleop
  - You will define and execute performance tests for important functionality
- You do not need to create highly detailed test plans, but you do need to focus your efforts:
  - Determine important performance criteria
  - Identify why those criteria are important
  - Define the metrics for the performance
  - Define the tests you will perform

Identify system metrics

- Examples of system metrics:
  - Average time to deliver a pizza
  - Percentage of pizzas successfully delivered
  - Average time to move Gompei
  - Etc., etc.

Ability to pick up pizzas off the ground

Value: Picking up pizzas gains significant points in teleop. We want to be confident that we can pick up a pizza in the CDR.

Metric: Percentage of pizzas picked up.

Test: Starting with 10 pizzas spread out on the arena, drive in teleop to each one and attempt to pick it up. If the pizza is picked up within 10 seconds, consider that a success.

Speed of aerial delivery

Value: Aerial delivery is typically the last action in teleop. We want to know when we need to begin the maneuver.

Metric: Average and standard deviation of the time taken to perform the maneuver.

Test: Starting with the BaseBot near the pizzeria, time how long it takes to complete the aerial delivery. Repeat the procedure 20 times.

Your turn: Define a useful test for your BaseBot

Be prepared to describe it to the class.

Strategies for Troubleshooting

Integrate incrementally

- Piece-wise integration with testing “as you go” can help ease the integration phase.
- Helps identify unintended consequences.
- Be methodical in your assembly/integration: Pick one thing to do at a time and focus on it.
- Work together as a team. Everyone needs to know where you are.
- Avoid assigning diverse tasks in parallel: “I’ll attach the arm while you program the ultrasonic sensor.”

Used focused tests to “divide and conquer”

- For example: Is it hardware or software?
  - Use a simple program to determine if the motors are moving.
  - Check the screen on the brain to see if the ultrasonic is producing output.

“It has to be the code!”

“It has to be the wiring!”

“It has to be the sensor!”

“It has to be the motor!”

Treat each test as an experiment

- Start with a hypothesis: What is supposed to happen?
- Test the hypothesis: What actually happens?

- Try to focus as specifically as you can

Record your efforts

- Keep track of what you have tested. This will not only help you avoid repetition, but also help your teammates think about what to try next.
- Keep track of system modifications (code, wiring, physical components). It’s never fun to have to backtrack to a previous (but working) state, but it’s even less fun when you don’t know what that state was!

Include observables in your design

- Test points in an electronic circuit
  - Night light not working as expected? Test the voltages at the op-amp connections to determine if it's an input or the output.
- Output statements for your state machine
  - "Pizza delivered! New destination: pizzeria."
  - "Moving arm to floor 2."

Design for adjustability

- Use potentiometers in an electronic circuit (e.g., the Braitenberg vehicle)
- Use parameters in your code:

```
#define FLOOR2_MOTOR_POS 1456

int lightDarkThreshold = 2233;
```

Keep the right attitude

- **Avoid troubleshooting alone**: Discussing your steps helps you think about what you're doing and what you've missed.
- **Avoid doing more harm than good**: The number of times I've heard someone say, "I thought maybe switching the wires would fix it."
- **Take a break**: Your system will be fine for ten minutes on its own and you can come back ready to try the next test