971 Software Architecture

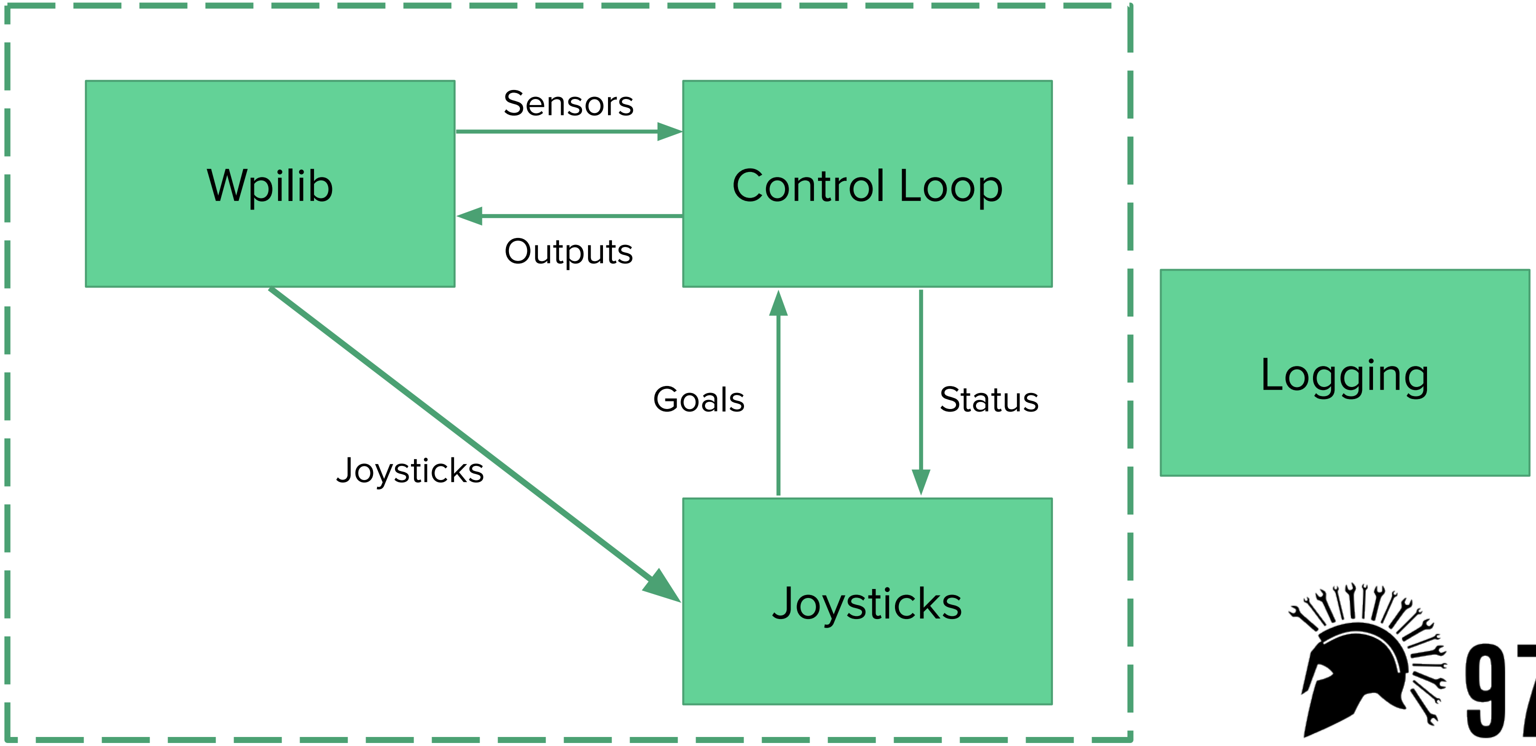
Tyler Chatow

971
SPARTAN ROBOTICS

Introduction

971 SPARTAN ROBOTICS

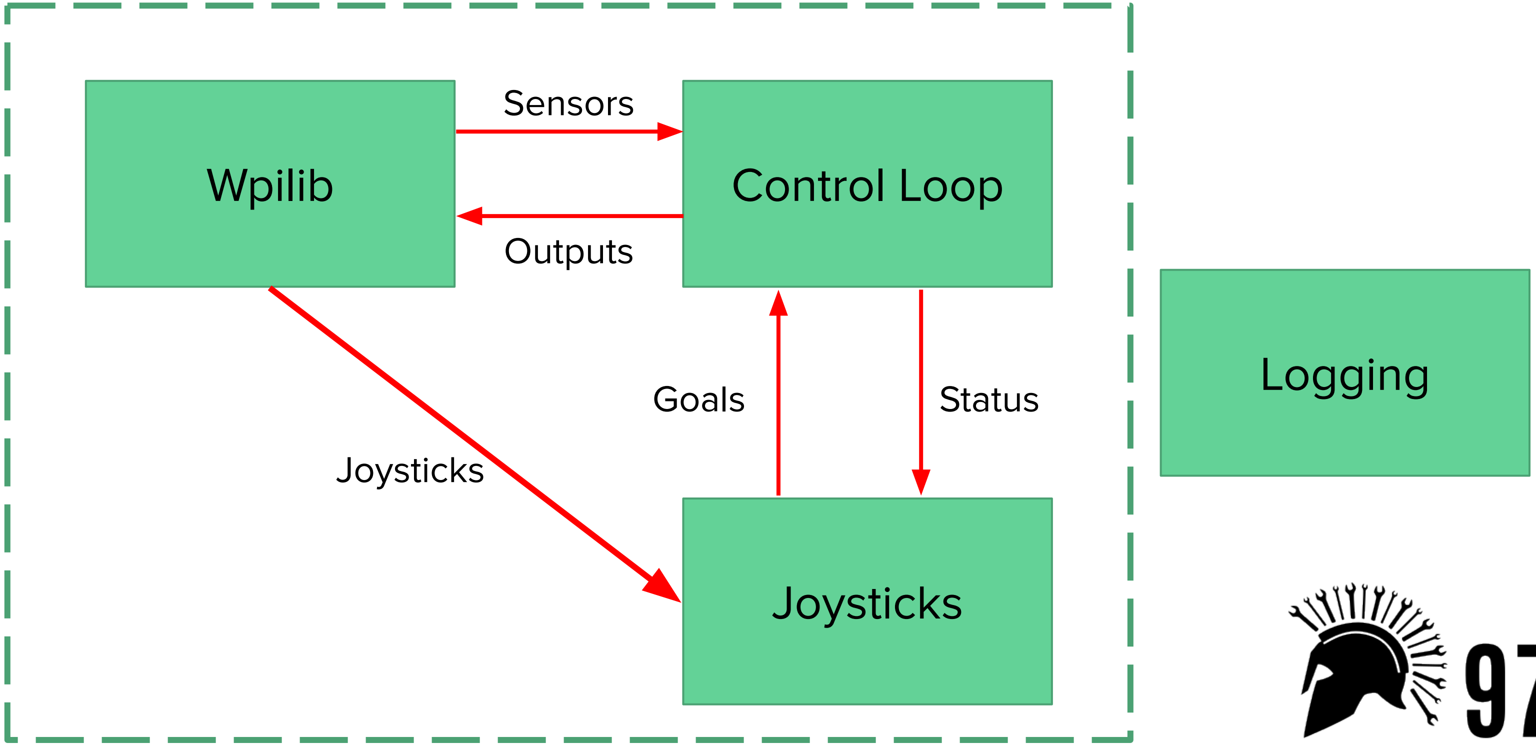
Code Stack Layout



Primary Components

- Messages - Standardized data structure for passing data between processes
- Wpilib Interface - Consistent and debuggable calls to the Roborio
- Control Loops - The core of a high performing robot system
- Testing Infrastructure - Ensure reasonable results before running on the robot
- Logging & Replays - Debugging robot behavior

Message Bus



971 SPARTAN ROBOTICS

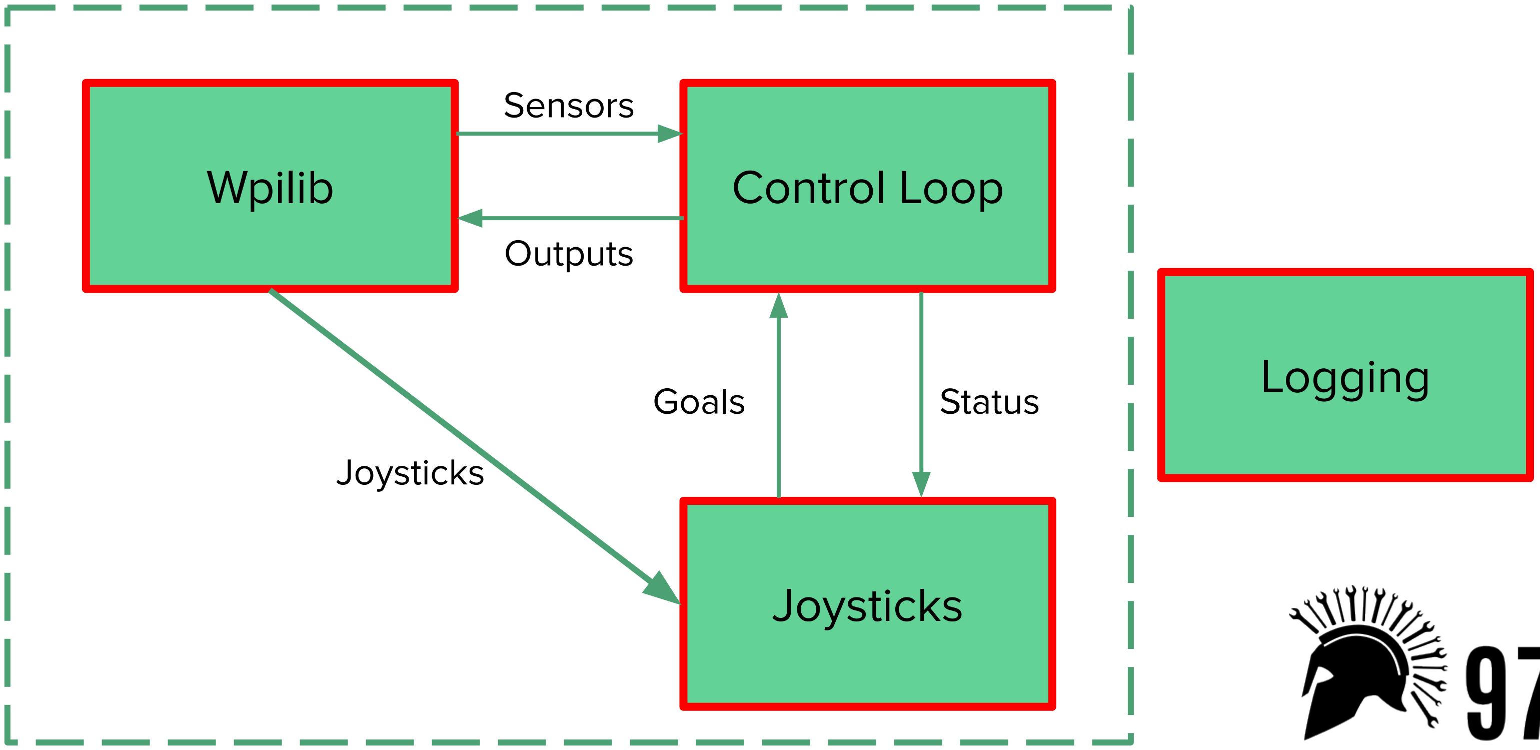
Message Bus

- Broadcast pub-sub
- Standardized access to the data of a given subsystem
- Provides an overview of the properties at a glance
- Ensures a single data state accessible from any process
- Typically 4 Types of Messages: Position -> Status -> Goal -> Output
- Abstracts interaction with a subsystem, can use without knowing implementation
- Queues are global and available to all processes

Example Message

```
queue_group SuperstructureQueue {
  implements aos.control_loops.ControlLoop;

  message Goal {
       // Meters, 0 = lowest position - mechanical hard stop,
       // positive = upward
       .frc971.control_loops.StaticZeroingSingleDOFProfiledSubsystemGoal elevator;

       // 0 = linkage on the sprocket is pointing straight up,
       // positive = forward
       .frc971.control_loops.StaticZeroingSingleDOFProfiledSubsystemGoal intake;

       // 0 = Straight up parallel to elevator
       // Positive rotates toward intake from 0
       .frc971.control_loops.StaticZeroingSingleDOFProfiledSubsystemGoal wrist;

       // Distance stilts extended out of the bottom of the robot. Positive = down.
       // 0 is the height such that the bottom of the stilts is tangent to the
       // bottom of the middle wheels.
       .frc971.control_loops.StaticZeroingSingleDOFProfiledSubsystemGoal stilts;

       // Positive is rollers intaking inward.
       double roller_voltage;

       SuctionGoal suction;
  };
}
```

Process Separation



971 SPARTAN ROBOTICS

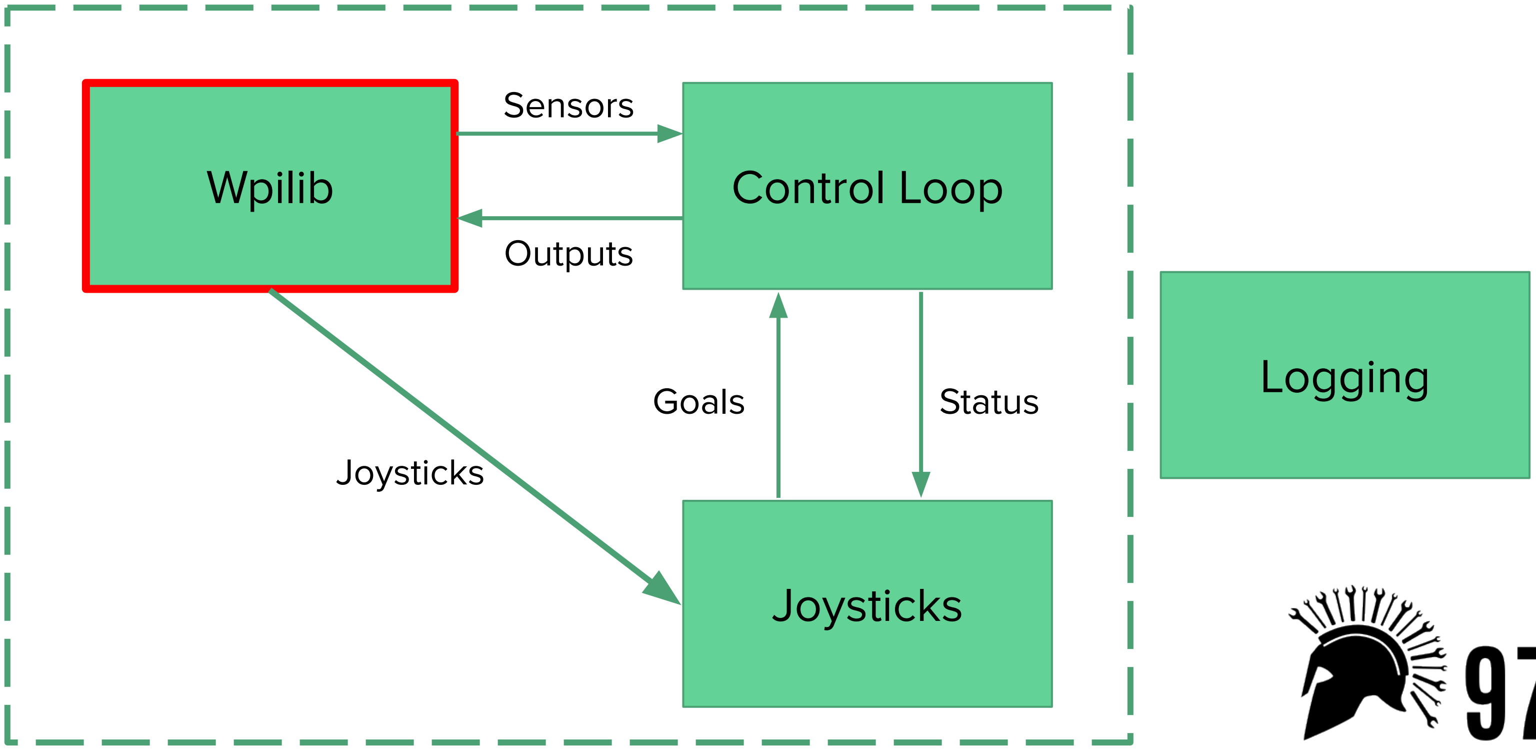
Process Separation

- Each system is separated into a separate process, which communicate via messages
- Messages stored in a queue in shared memory that can be accessed by all processes
- Individual systems can be estopped automatically when an inconsistency is detected
- Crashes don’t bring down the entire robot
- Processes are separated for higher performance

Message Bus (Memory Management)

- Block of memory allocated for all queues in a tmpfs
- Queues are stored similar to a ring buffer
- Once a control loop is done writing its output queue, it sends it
  - Grabs a mutex to the queue, gets the current index, overwrites the oldest message, increments the index
- Other processes read the index and access the data in the queue

Wpilib Interface



971 SPARTAN ROBOTICS

Wpilib Interface

- Single process that handles (nearly) all communication with wpilib
- Reads values from output messages set by control loops
- Writes sensor values to position message
- Can be updated for changes to Wpilib without modifying any other code
- One source of truth for all communication with the robot

What could go wrong with this?

```
int RunAutonomous() {
      // Drive straight
      SetVoltage(talonLeftIdx, 12.0);
      SetVoltage(talonRightIdx, 12.0);
      sleepMillis(2000);
      // Turn
      SetVoltage(talonRightIdx, 10.0);
      sleepMillis(500);

      // Drive straight and raise elevator
      SetVoltage(talonRightIdx, 12.0);
      SetVoltage(elevatorIdx, 12.0);
      sleepMillis(1000);

      // Stop
      SetVoltage(talonLeftIdx, 0.0)
      SetVoltage(talonRightIdx, 0.0)
      SetVoltage(elevatorIdx, 0.0);
      ReleaseSuction();
}
```

What happens when the starting platform is slicker?

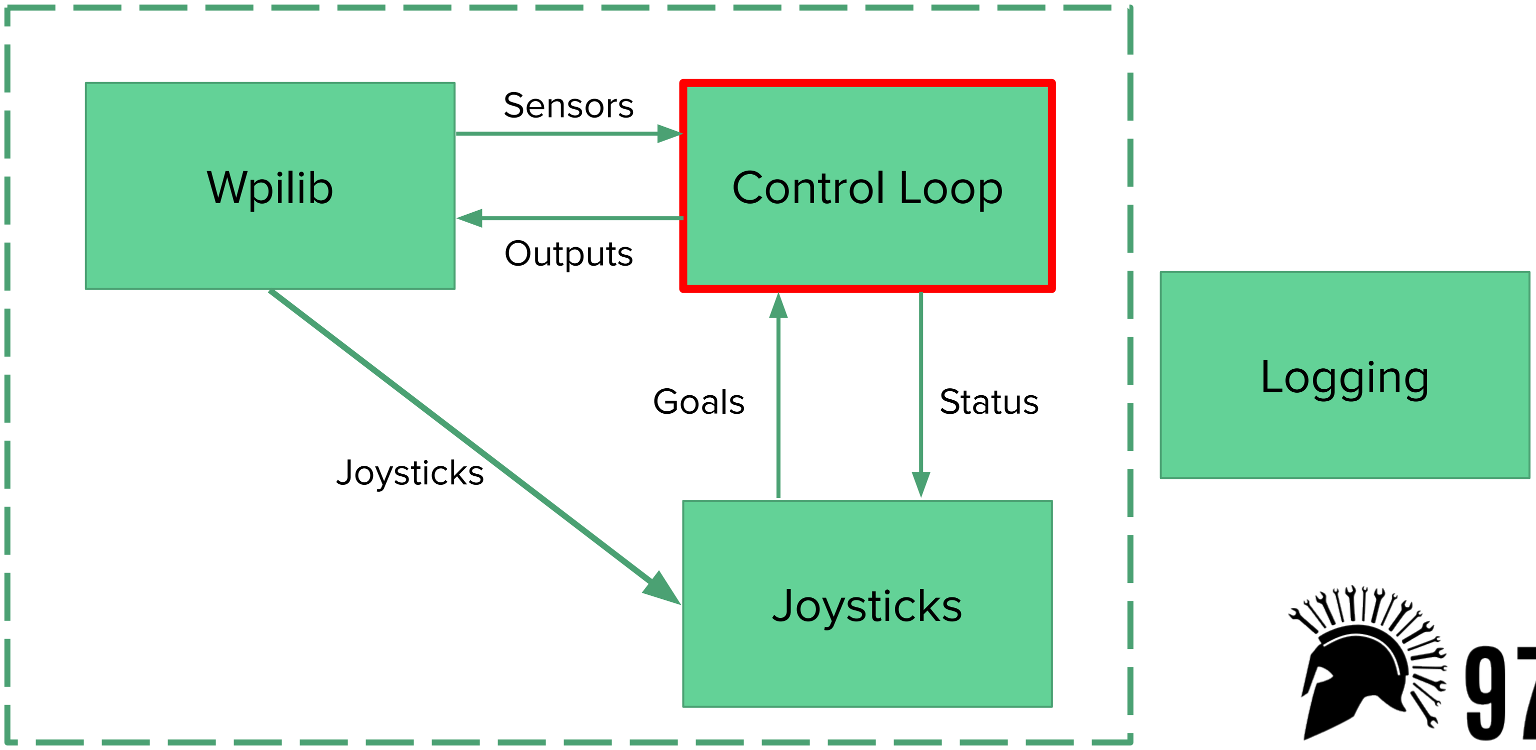
What if the battery run on the robot is old?

What if one of your motors is wearing out?

What if the robot isn't aligned precisely on the field?

SPARTAN ROBOTICS 971

Control Loops



971 SPARTAN ROBOTICS

What is a control loop?

“A control loop is the fundamental building block of industrial control systems. It consists of all the physical components and control functions necessary to automatically adjust the value of a measured process variable (PV) to equal the value of a desired set-point (SP). It includes the process sensor, the controller function, and the final control element (FCE) which are all required for automatic control.”

What is a control loop?

- Handles the process of adjusting a system from its current state to a target state
- Responds to real world inputs
- Moving an elevator from height x to y
- Collision avoidance between an extendable intake and an arm
- Driving in a fixed direction for a given distance
- Actuating a solenoid to clamp when a game piece is detected

What is a control loop?

- Ensures consistent robot behavior in varying conditions
- Responds to inputs to produce outputs
- Using an existing control system is approachable and “Plug and play”
- Combines data from sensors with goal to determine output

971 SPARTAN ROBOTICS

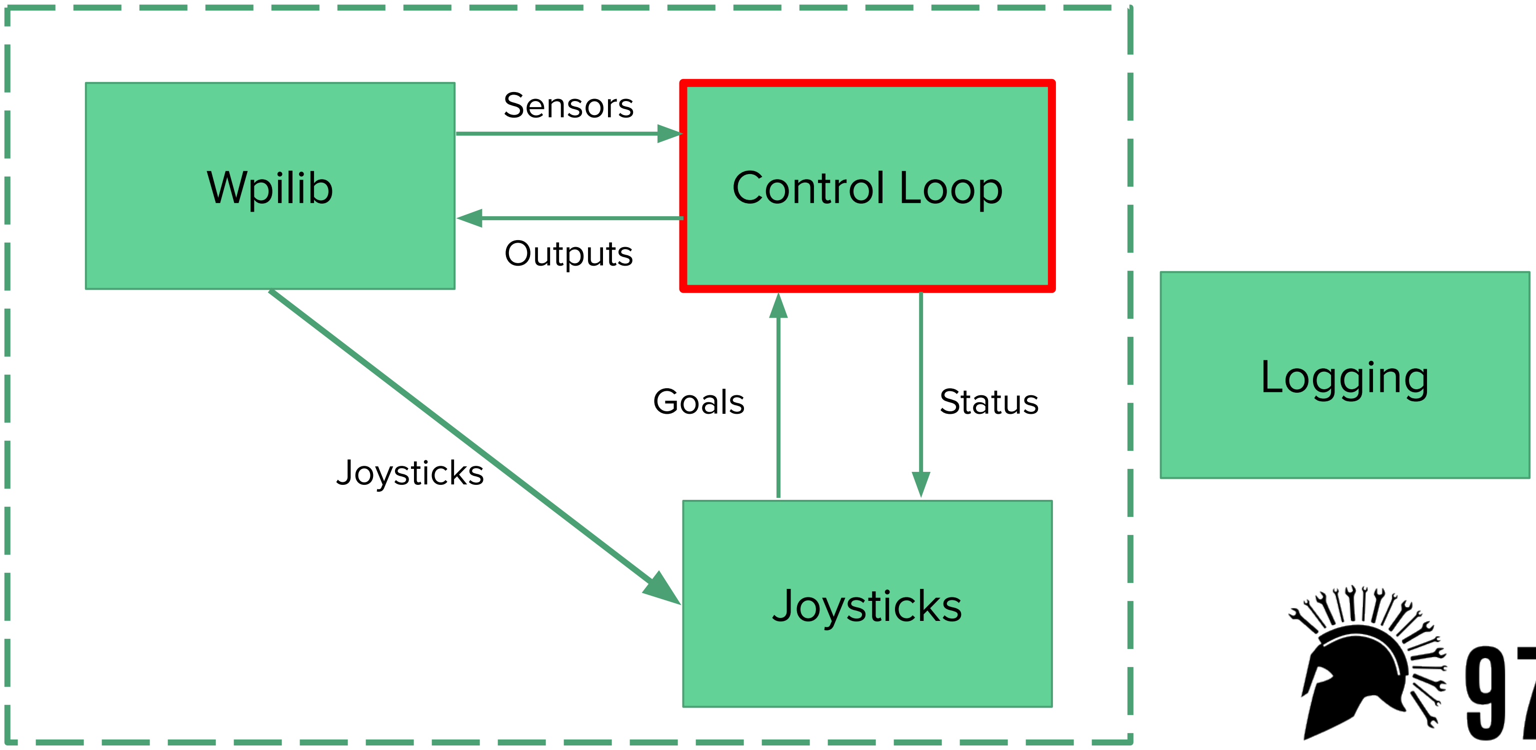
Simple Example

```
class Drivetrain {

  void Iterate() {
    voltage_left = abs(target_distance - encoder_left) > 0.01 ? sign(target_distance - encoder_left) * 12.0 : 0.0;
    voltage_right = abs(target_distance - encoder_right) > 0.01 ? sign(target_distance - encoder_right) * 12.0 : 0.0;
  }

  void SetDistanceGoal(double distance) {
    // Set distance as an offset from the current encoder positions
    target_distance_left = distance + encoder_left;
    target_distance_right = distance + encoder_right;
  }

  bool GoalReached() {
    return abs(encoder_left - target_distance_left) <= 0.01 &&
           abs(encoder_right - target_distance_right) <= 0.01;
  }
}
```

Better?

```
int RunAutonomous() {
      // Drive straight
      SetDistanceGoal(2.0) // drive in meters
      while(!GoalReached()) {Iterate();}

      // Turn 45 degrees right
      SetTurnGoal(-M_PI / 4);
      while(!GoalReached()) {Iterate();}

      // Drive forward and raise elevator
      SetDistanceGoal(1.0);
      SetElevatorHeight(1.5);
      while(!GoalReached()) {Iterate();}

      ReleaseSuction();
}
```

Code Path


Wpilib
Sensors
Outputs
Control Loop
Goals
Status
Joysticks
Joysticks
Logging

2019 Example

```
void RunIteration() override {

     auto superstructure_message = superstructure_queue.position.MakeMessage();

     // Elevator
     CopyPosition(elevator_encoder_, &superstructure_message->elevator,
                  Values::kElevatorEncoderCountsPerRevolution(),
                  Values::kElevatorEncoderRatio(), elevator_pot_translate,
                  false, values.elevator.potentiometer_offset);

     superstructure_message.Send();
}
```

2019 Example

```
const ElevatorWristPosition kPanelForwardMiddlePos{0.75, M_PI / 2.0};

void HandleTeleop(const ::aos::input::driver_station::Data &data) override {
     auto new_superstructure_goal = superstructure_goal_sender_.MakeMessage();


     if (data.IsPressed(kRocketForwardMiddle)) {
          elevator_wrist_pos_ = kPanelForwardMiddlePos;
     }

     new_superstructure_goal->elevator.unsafe_goal =
          elevator_wrist_pos_.elevator;
     new_superstructure_goal->wrist.unsafe_goal = elevator_wrist_pos_.wrist;

     new_superstructure_goal.Send();
}
```

971
SPARTAN ROBOTICS

2019 Example

```
void Superstructure::RunIteration(const SuperstructureQueue::Goal *unsafe_goal,
                                   const SuperstructureQueue::Position *position,
                                   SuperstructureQueue::Output *output,
                                   SuperstructureQueue::Status *status) {

  collision_avoidance_.UpdateGoal(status, unsafe_goal);
  elevator_.set_min_position(collision_avoidance_.min_elevator_goal());

...

  elevator_.Iterate(unsafe_goal != nullptr ? &(unsafe_goal->elevator) : nullptr,
                    &(position->elevator),
                    output != nullptr ? &(output->elevator_voltage) : nullptr,
                    &(status->elevator));
}
```

2019 Example

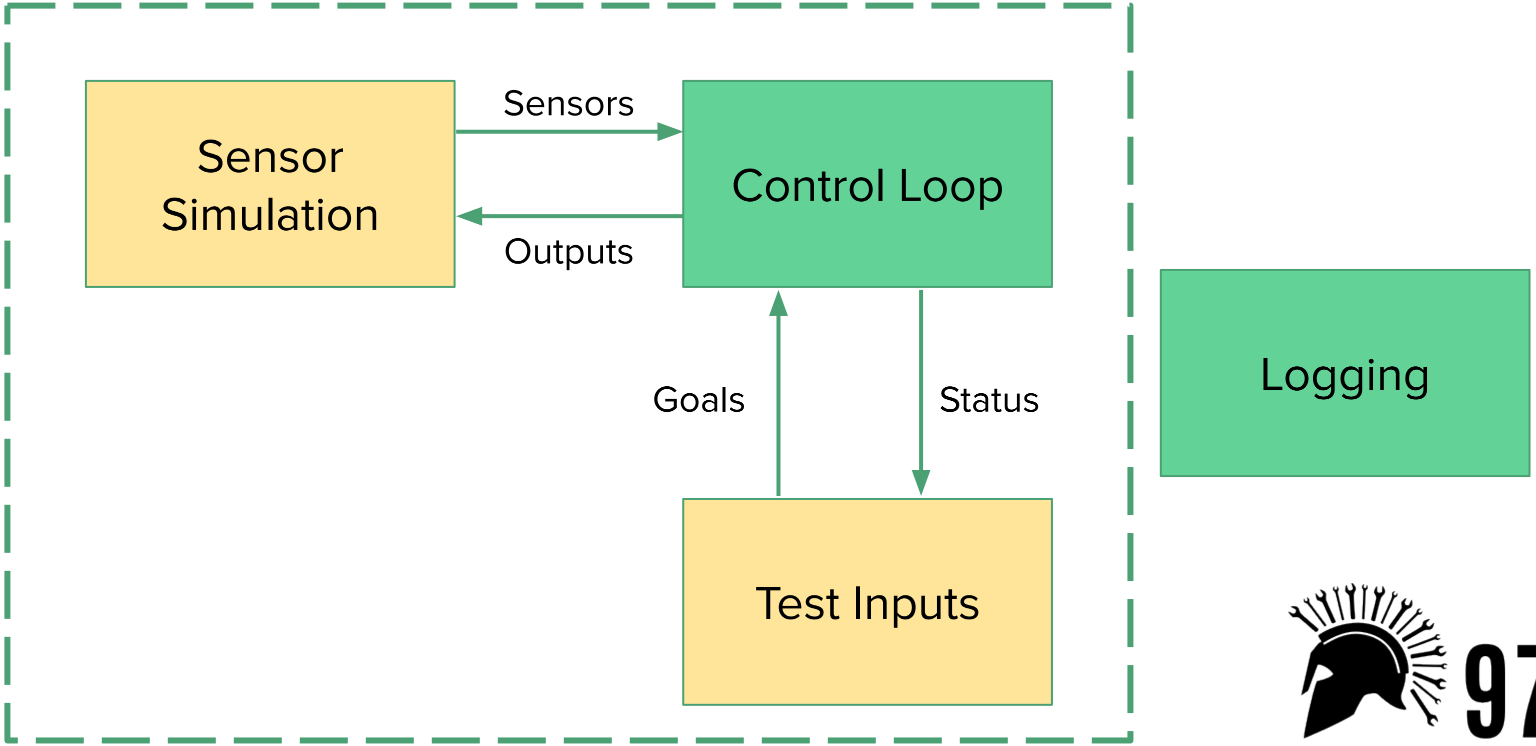
```
void Write(const SuperstructureQueue::Output &output) override {
  elevator_victor_->SetSpeed(::aos::Clip(output.elevator_voltage,
                                        -kMaxBringupPower,
                                        kMaxBringupPower) / 12.0);
}
```

971 SPARTAN ROBOTICS

Testing Infrastructure

- Utilize GTest tools to automate testing C++ code
- Continuous testing ensures all tests pass before code is merged
- Test Driven Development - tests are written to check for correct behavior, then code written to satisfy those tests
- Tests for both API - Control loops, Zeroing, Messages, AOS; and year specific code - Elevator, wrist, and intake collision, suction

Testing Simulation



How?

- All data is passed through messages
- Control loops can be run to interact with testing and simulation instead of wpilib interface
- Sensor responses to voltage outputs are simulated
- Layers of abstraction allow for swapping out components as necessary for testing; code not directly linked to voltage outputs
- C++ Templates
- Logging of all messages

971
SPARTAN ROBOTICS

Logging

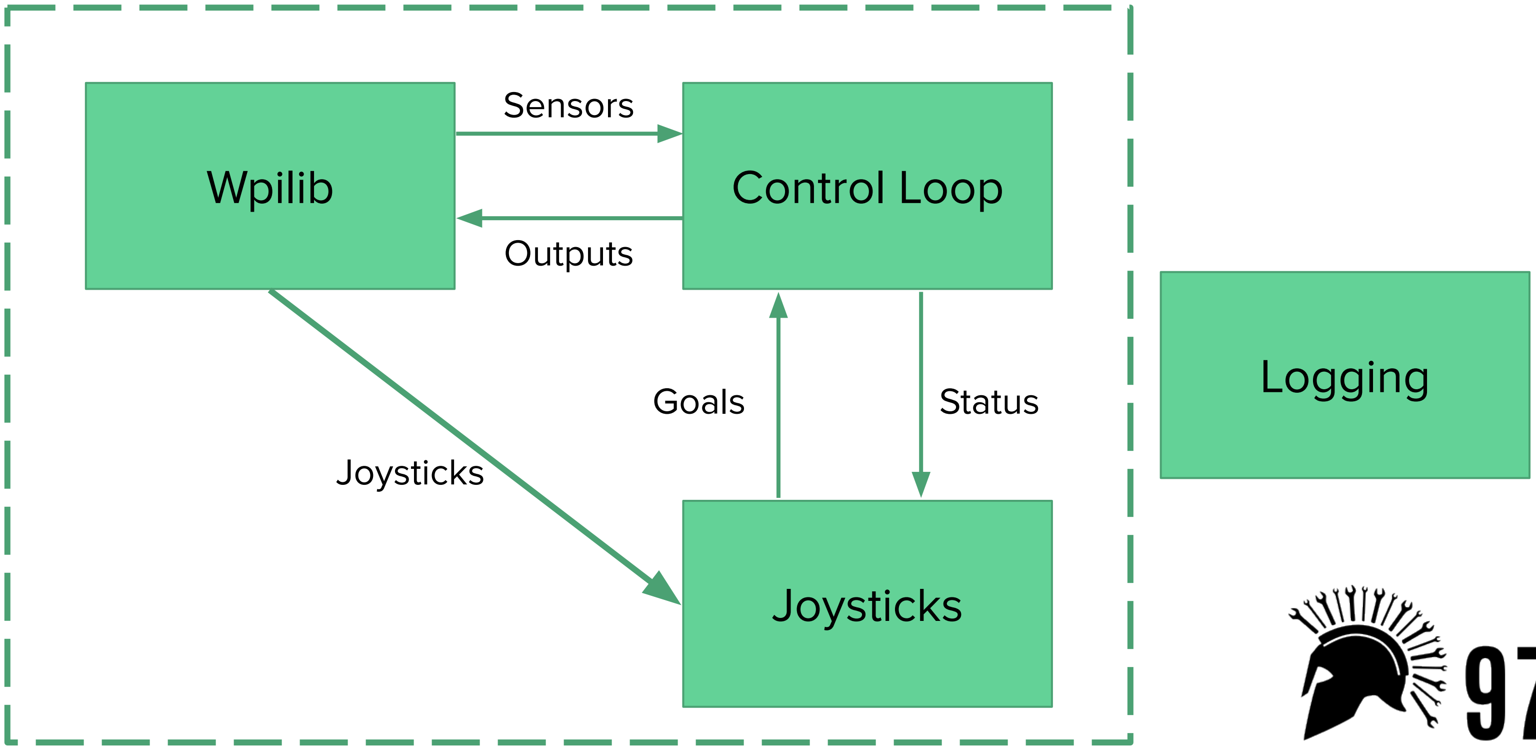
- Record all messages passed between queues, 200Hz
- For testing, positions can be plotted to determine what happened, compared to the expected output
- Data logged on a flash drive while the robot is on, both before and during a match
- Log streamer can view data while the robot is on
- Logged data can be replayed or analyzed to determine the cause of a problem, and compared with match video

971 SPARTAN ROBOTICS

Thank You!

971
SPARTAN ROBOTICS

Questions?


Wpilib
Sensors
Outputs
Control Loop
Goals
Status
Joysticks
Joysticks
Logging