A simple and modern dashboard for FRC

Team 353 – The POBots

Driver Dashboard Options

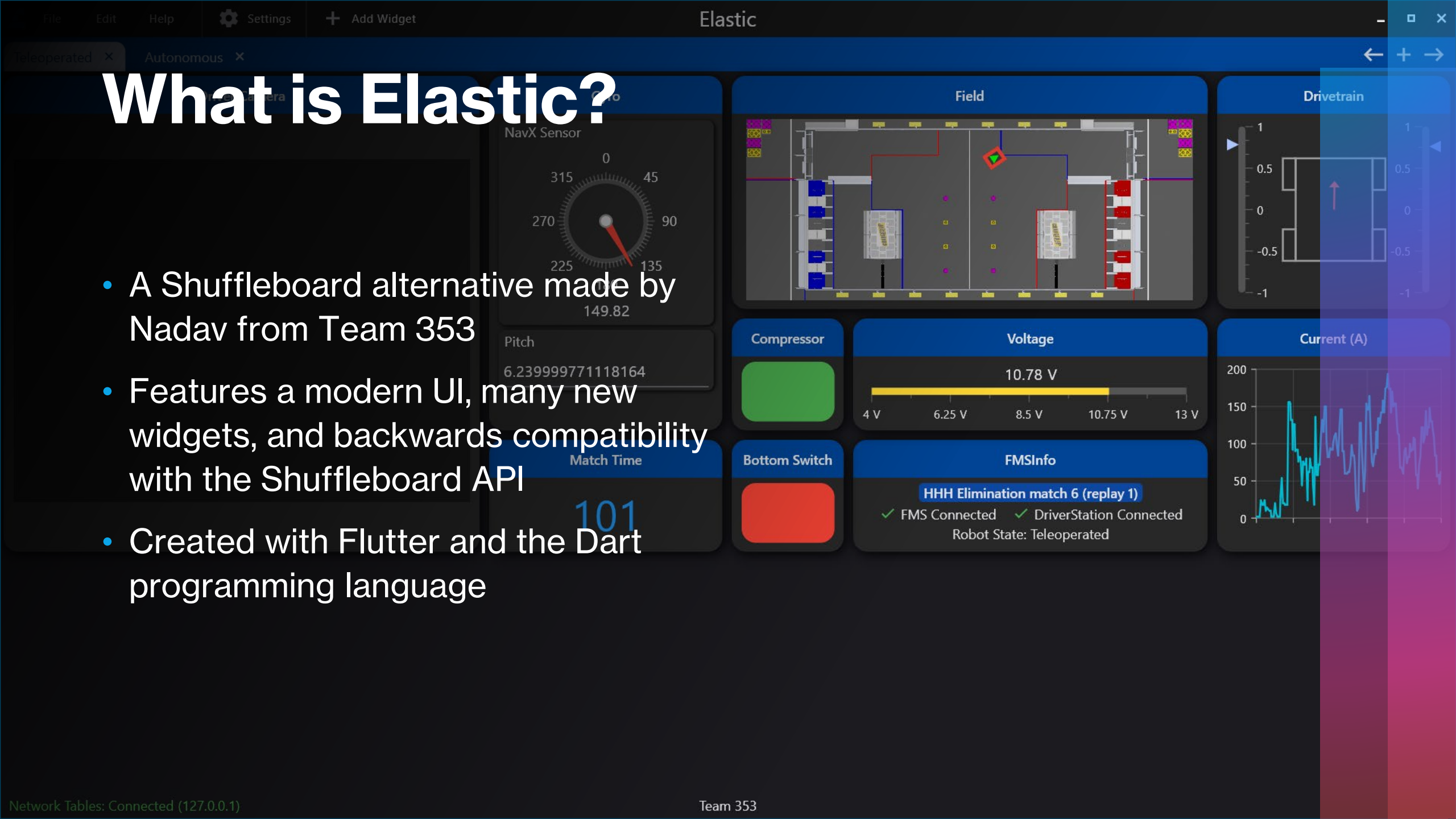
- SmartDashboard
  - Pros: Tons of different widgets
  - Cons: Lacks customization, no longer being maintained by WPILib
- Shuffleboard
  - Pros: Ability to arrange cards in a grid, code generated layouts
  - Cons: Extremely slow, no longer maintained

Problems with Shuffleboard

- No longer maintained by the creators of WPILib
- Very slow, leaks memory, and has many unpatched bugs
- Outdated: widgets such as swerve drive visualization do not exist
- UI is no longer elegant nor modern compared to today's standards

What is Elastic?

- A Shuffleboard alternative made by Nadav from Team 353
- Features a modern UI, many new widgets, and backwards compatibility with the Shuffleboard API
- Created with Flutter and the Dart programming language



Reason for the Name

1. Elastic is an alternative to Shuffleboard, and shuffleboard pucks collide in perfectly elastic collisions
2. Elastic materials are very flexible

Video Demo

Why Flutter?

- Native multiplatform support, makes code easier to maintain
- Its material framework is very easy to work with and makes apps much easier to look at
- Lots of built in widgets, along with many open-source packages for custom features
- Very fast

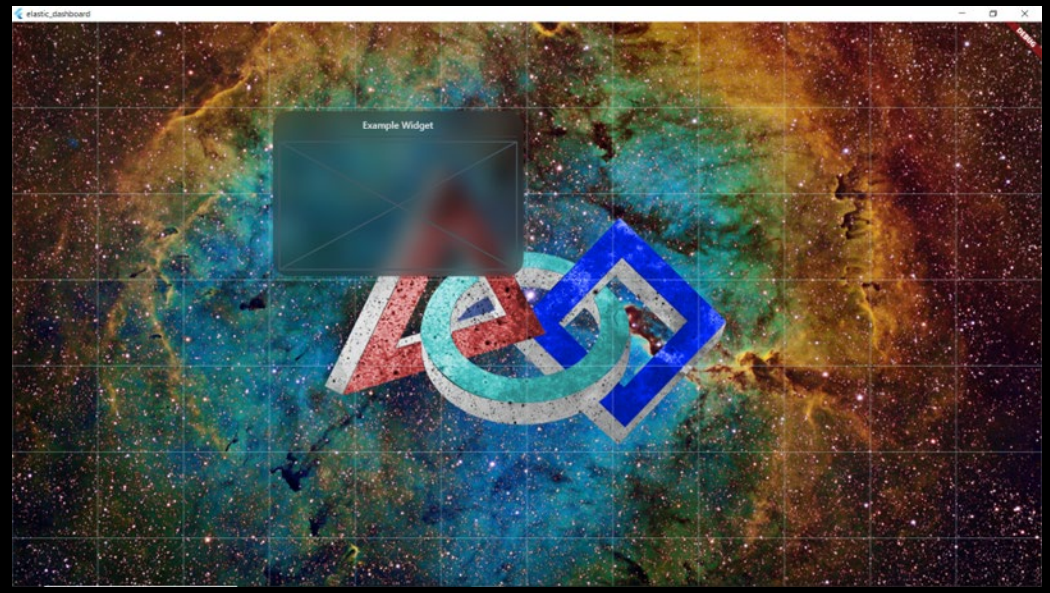
Flutter

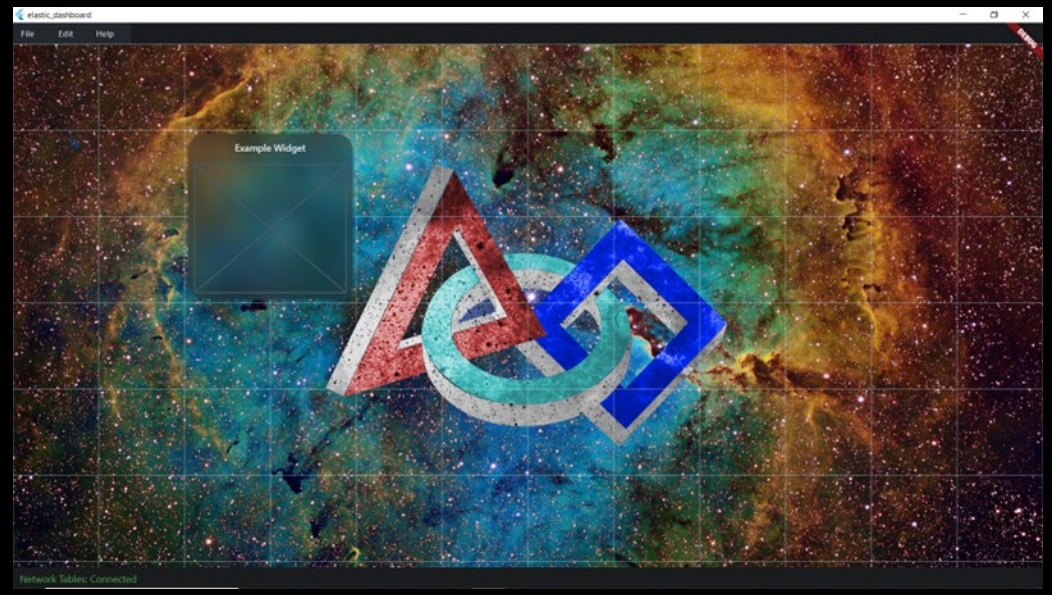
Goals

- Intuitive and modern design
- Fast and efficient for both the computer and the network
- Well documented
- Easy for teams to adapt to

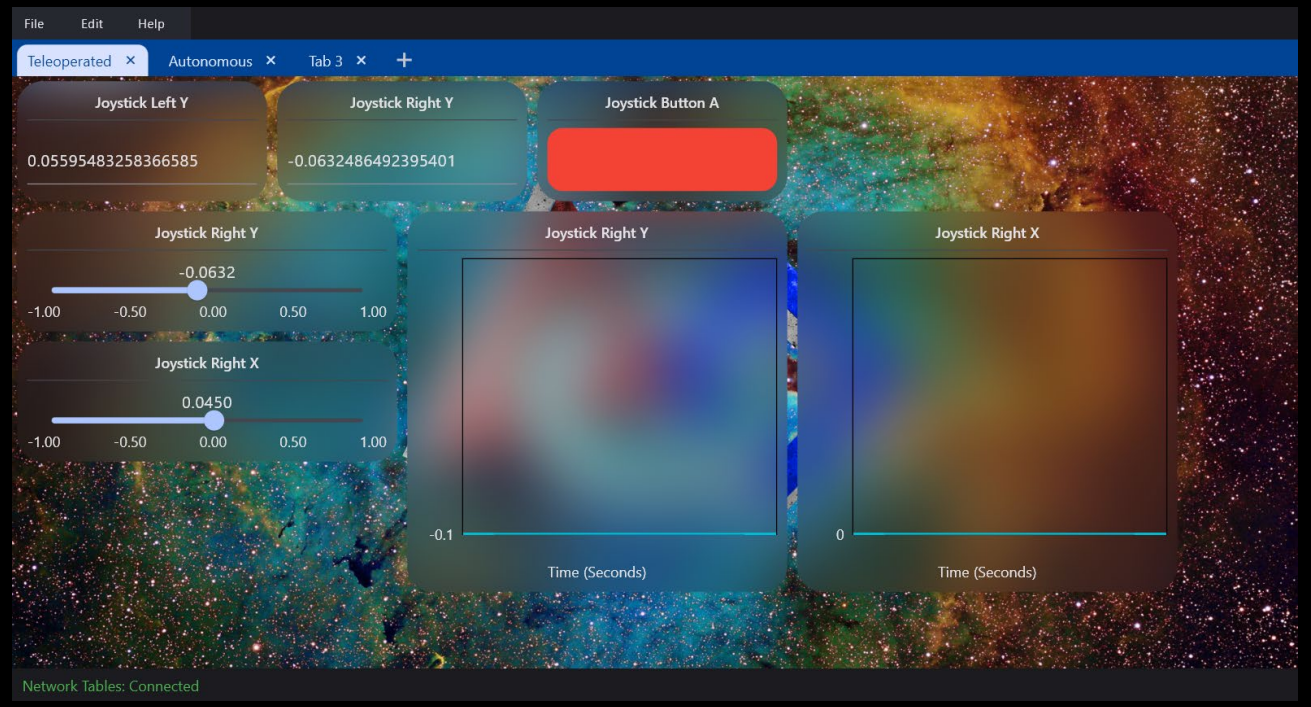
Early Prototypes

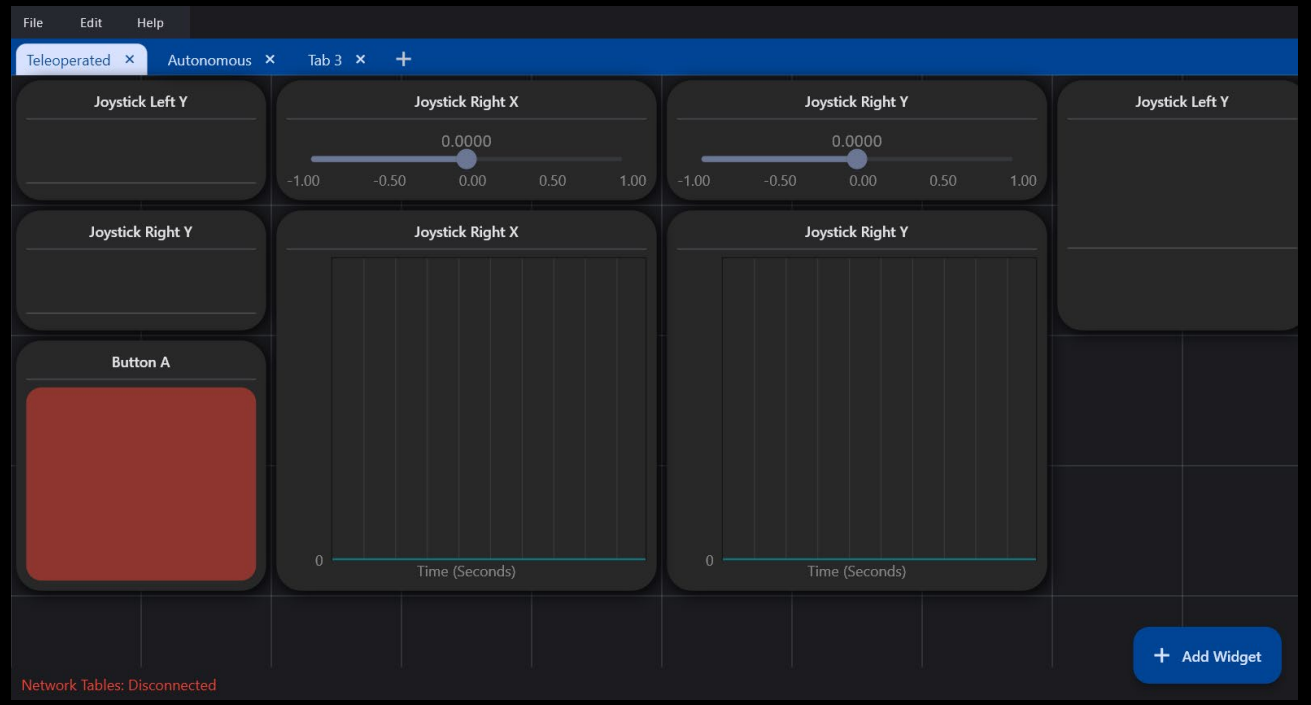
- Tested dragging and snapping boxes to a grid
- Made sure that a reliable connection to Network Tables was possible





UI Design Iterations





Connecting to Network Tables

- Borrowed and modified a library written by Team 3015
- Connects to the robot over a websocket channel with the Network Tables protocol

Network Tables Terms

- Topic – The name of a data entry
- Subscription – A receiver for a topic's data
- Client – A program that connects to Network Tables

Code for Initializing Connection

elastic_dashboard - nt4.dart

```dart
_clientId = Random().nextInt(99999999);

String mainServerAddr = 'ws://$serverBaseAddress:5810/nt/elastic';

_mainWebsocket = WebSocketChannel.connect(Uri.parse(mainServerAddr),
    protocols: ['networktables.first.wpi.edu']);

try {
  await _mainWebsocket!.ready;
} catch (e) {
  // Failed to connect... try again
  Future.delayed(const Duration(seconds: 1), wsConnect);
  return;
}
```

elastic_dashboard - nt4.dart

```dart
_mainWebsocket!.stream.listen(
  (data) {
    // Prevents repeated calls to onConnect and reconnecting after changing ip addresses
    if (!_serverConnectionActive &&
        mainServerAddr.contains(serverBaseAddress)) {
      lastAnnouncedValues.clear();

      for (NT4Subscription sub in _subscriptions.values) {
        sub.currentValue = null;
      }

      _serverConnectionActive = true;

      onConnect?.call();
    }
    _wsOnMessage(data);
  },
  onDone: _wsOnClose,
  onError: (err) {
    if (kDebugMode) {
      print('NT4 ERR: $err');
    }
  },
);
```

Code for Streaming Incoming Data

elastic_dashboard - nt4.dart

```dart
if (method == 'announce') {
  NT4Topic? currentTopic;
  for (NT4Topic topic in _clientPublishedTopics.values) {
    if (params['name'] == topic.name) {
      currentTopic = topic;
    }
  }

  NT4Topic newTopic = NT4Topic(
      name: params['name'],
      type: params['type'],
      id: params['id'],
      pubUID: params['pubid'] ?? (currentTopic?.pubUID ?? 0),
      properties: params['properties']);
  announcedTopics[newTopic.id] = newTopic;

  for (final listener in _topicAnnounceListeners) {
    listener.call(newTopic);
  }
} else if (method == 'unannounce') {
  NT4Topic? removedTopic = announcedTopics[params['id']];
  if (removedTopic == null) {
    if (kDebugMode) {
      print(
          '[NT4] Ignorining unannounce, topic was not previously announced');
    }
    return;
  }
  announcedTopics.remove(removedTopic.id);
} else if (method == 'properties') {
} else {
  if (kDebugMode) {
    print('[NT4] Ignoring text message - unknown method $method');
  }
  return;
}
}
```

elastic_dashboard - nt4.dart

```dart
var u = Unpacker.fromList(data);

bool done = false;
while (!done) {
  try {
    var msg = u.unpackList();

    int topicID = msg[0] as int;
    int timestampUS = msg[1] as int;
    var value = msg[3];

    if (topicID >= 0) {
      NT4Topic topic = announcedTopics[topicID]!;
      lastAnnouncedValues[topic.name] = value;
      for (NT4Subscription sub in _subscriptions.values) {
        if (sub.topic == topic.name) {
          sub.updateValue(value);
        }
      }
    } else if (topicID == -1) {
      _rttHandleRecieveTimestamp(timestampUS, value as int);
    } else {
      if (kDebugMode) {
        print('[NT4] ignoring binary data, invalid topic ID');
      }
    }
  } catch (err) {
    done = true;
  }
}
```

Saving Bandwidth

- Only subscribe to Network Tables topics that are necessary for displaying information the dashboard needs
- Share Network Tables subscriptions between widgets
- Once subscriptions are no longer used by widgets, unsubscribe to conserve bandwidth

Code for Subscription Instance Counting

elastic_dashboard - nt4.dart

```dart
NT4Subscription subscribe(String topic, [double period = 0.1]) {
  NT4Subscription newSub = NT4Subscription(
    topic: topic,
    uid: getNewSubUID(),
    options: NT4SubscriptionOptions(periodicRateSeconds: period),
  );

  if (_subscribedTopics.contains(newSub)) {
    NT4Subscription subscription = _subscribedTopics.lookup(newSub)!;
    subscription.useCount++;

    return subscription;
  }

  newSub.useCount++;

  _subscriptions[newSub.uid] = newSub;
  _subscribedTopics.add(newSub);
  _wsSubscribe(newSub);

  if (lastAnnouncedValues.containsKey(topic)) {
    newSub.updateValue(lastAnnouncedValues[topic]);
  }

  return newSub;
}
```

elastic_dashboard - nt4.dart

```dart
void unSubscribe(NT4Subscription sub) {
  sub.useCount--;

  if (sub.useCount <= 0) {
    _subscriptions.remove(sub.uid);
    _subscribedTopics.remove(sub);
    _wsUnsubscribe(sub);
  }
}
```

Displaying Data from Network Tables

- Stream asynchronous data from network tables
- Rebuild widgets as new data is updated
- Each widget uses its own data stream to improve performance

Displaying Data Example: Boolean Box

elastic_dashboard - boolean_box.dart

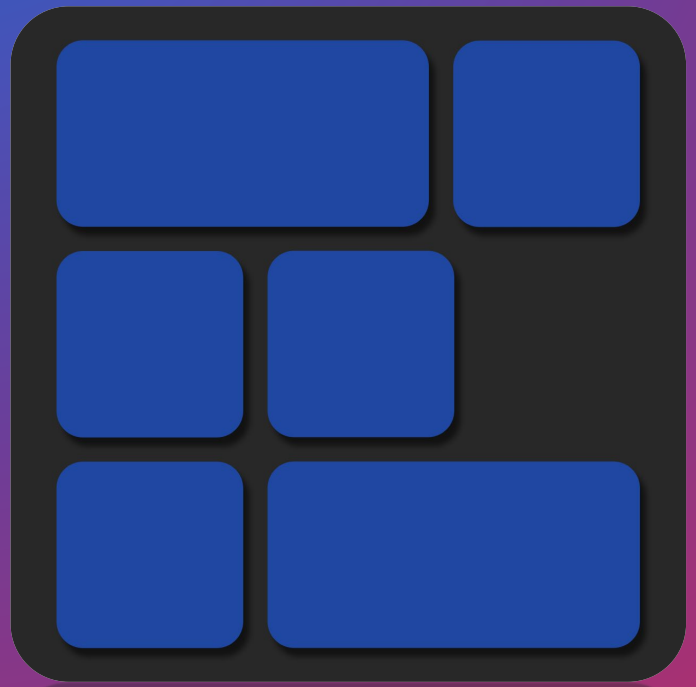
```dart
@override
Widget build(BuildContext context) {
  notifier = context.watch<NT4WidgetNotifier?>();

  return StreamBuilder(
    stream: subscription?.periodicStream(),
    initialData: nt4Connection.getLastAnnouncedValue(topic),
    builder: (context, snapshot) {
      bool value = tryCast(snapshot.data) ?? false;

      return Container(
        decoration: BoxDecoration(
          borderRadius: BorderRadius.circular(15.0),
          color: (value) ? trueColor : falseColor,
        ),
      );
    },
  );
}
```

Feedback and Testing

- Tested by several teams at different offseason events, all reporting a positive experience with suggestions on how to continue improving it
- We appreciate all feedback and are open to new ideas!



Thank you!
Any questions?

Download Link + Source Code:
www.github.com/Gold872/elastic-dashboard

Chief Delphi Announcement Thread:
www.chiefdelphi.com/t/440750

Source Code

Chief Delphi Thread