

Working with Buttons - Clicker Racing Game

Case Study

Contents

1. Overview	2
2. Hardware Setup	3
3. Background Information	5
4. Circuit Design – step by step instructions	5
4.1 Description	5
4.2 Step by step instructions	6
4.3 Assembled Circuit	7
5. Software Setup - step by step instructions	8
5.1 Software Setup in Waveforms	8
5.2 Data Visualization Using NI LabVIEW WaveForms Toolkit	9
6. Clicker Racing Game in LabVIEW	10
6.1 Description	10
6.2 How to open and run the project:	11

1. Overview

Short description: This **case study** explores how push buttons can be used as digital input devices to interact with a simple game application. Participants learn how button presses are detected through digital read channels and how these signals can be used to control events in real time. As a demonstration, a Clicker Racing Game is used, where each button press advances a player and creates a competitive interactive experience. The Analog Discovery Product Line (AD3 or AD Studio / Studio MAX, hereby **Device**) is used to read button states and transfer them to the game logic, demonstrating the practical use of digital input channels in an engaging way.

Objectives

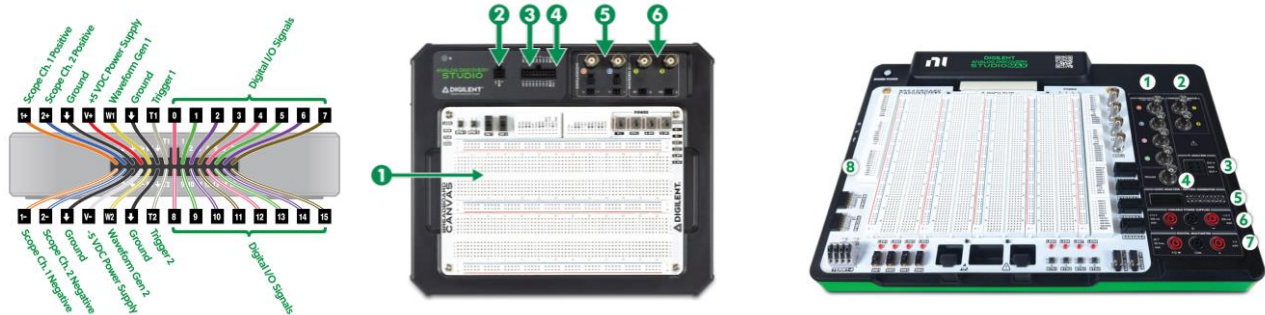
- Understand the basic operating principle of push buttons as digital input devices.
- Learn how to use digital read channels to detect button states.
- Build and test a simple button-based input setup using the **Device**.
- Explore digital input concepts through a Clicker Racing Game demonstration.

Software & Hardware Requirements

Software		Version
Digilent WaveForms (Runtime + SDK)		Latest (64-bit)
NI LabVIEW		2025 64-bit
WaveForms Toolkit for LabVIEW		Latest
Hardware		Quantity*
Digilent Analog Discovery Product Line		
Option 1	Digilent Analog Discovery 3	1
Option 2	Digilent Analog Discovery Studio	1
Option 3	Digilent Analog Discovery Studio MAX	1
List of Components		Quantity*
Canvas Breadboard		1
Push Buttons		2
Resistor 10kΩ		2
Jumper wires		-

2. Hardware Setup

This section describes the **Digilent Analog Discovery Product Line** hardware setup for **Case studies**. Before assembling the measurement circuit, it is essential to understand the pinout and available connections of the **Devices**. **Figure 1** highlights the pinouts and instrumentations of the Digilent Analog Discovery product line.¹



a. Analog Discovery 3

- Power Supplies
- Logic Analyzer and Pattern Generator
- Triggers
- Oscilloscope Waveform Generator

b. Analog Discovery Studio

1. Breadboard Canvas
2. Power Supplies
3. Logic Analyzer and Pattern Generator
4. Triggers
5. Oscilloscope
6. Waveform Generator

c. Analog Discovery Studio MAX

1. Oscilloscope
2. Waveform Generator
3. IV Analyzer
4. Trigger
5. Logic Analyzer and Pattern Generator
6. Variable Power Supplies
7. Digital Multimeter
8. Breadboard Canvas

Figure 1. Digilent Analog Discovery Products Pinout and Instrumentations

Table 1 provides a common reference for physical channels' names across the **Digilent Analog Discovery product line**. The **Case studies** are designed to work with all supported devices, so the generic channels' names shown in the first row are used consistently in all circuit diagrams and wiring instructions.

Table 2 provides a common reference for device names and instrument identifiers used in NI LabVIEW with the WaveForms Toolkit across the **Digilent Analog Discovery product line**. The Case studies are designed to work with all supported devices; therefore, the generic instruments and

¹ For detailed electrical specifications, pin descriptions, and operating limits, always refer to the official datasheets of each device (Analog Discovery 3, Analog Discovery Studio, and Analog Discovery Studio MAX). The latest datasheets are available on the Digilent website:

- [Analog Discovery 3](#)
- [Analog Discovery Studio](#)
- [Analog Discovery Studio MAX](#)

channels identifiers shown in Table 2 are used consistently throughout the LabVIEW block diagrams and configuration steps.

Table 1 Physical Channels Mapping for Digilent Analog Discovery Devices

<i>Channel Device</i>	<i>DIO 0</i>	<i>DIO 1</i>	<i>DIO 2</i>	<i>DIO 3</i>
AD 3	0	1	2	3
AD Studio ²	DIO 0	DIO 1	DIO 2	DIO 3
AD Studio MAX ² above	LOGIC ANALYZER 0	LOGIC ANALYZER 1	LOGIC ANALYZER 2	LOGIC ANALYZER 3

Table 2 Device Names and Instruments Identifiers in NI LabVIEW (WaveForms Toolkit)

<i>Identifier Device</i>	<i>Device name</i>	<i>DIO 0-1</i>	<i>DIO 2-3</i>
AD 3	Discovery3	dig/0:1	dig/2:3
AD Studio	DiscoveryStudio		
AD Studio MAX	ADS_Max		

² For the Analog Discovery Studio and Analog Discovery Studio MAX, ensure the devices are powered by turning ON the main power switch before use.

3. Background Information

A push button is a simple electromechanical switch commonly used to provide digital input in electronic circuits. In this case study, two push buttons are used to generate logic-level signals that can be detected by digital input channels. The circuit is shown in Figure 2.

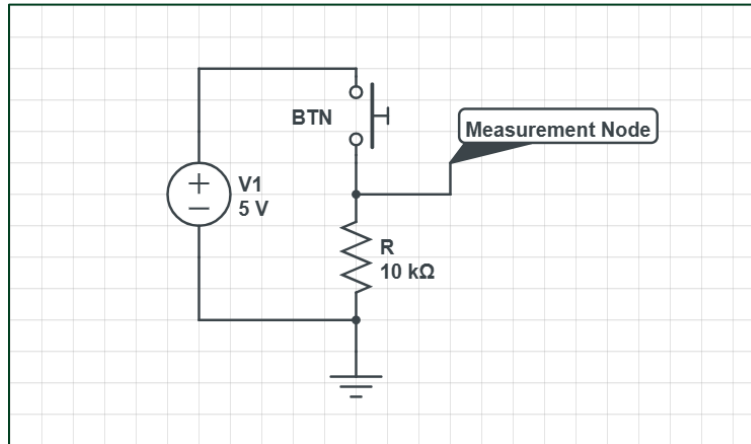


Figure 2. Push-Button Digital Input Circuit with Pull-Down Resistors

In the circuit, push button is connected in series with a 10 kΩ resistor. This arrangement limits current flow and helps establish a defined voltage level at the **Measurement Node** when the button is pressed. When the button changes state, the voltage at the corresponding node also changes, producing a transition that can be interpreted as a digital LOW or HIGH signal.

4. Circuit Design – step by step instructions

4.1 Description

The measurement circuit is based on a simple digital input configuration used for button state detection. Two push buttons, **BTN1** and **BTN2**, are each connected in series with a 10 kΩ resistor (**R1** and **R2**) referenced to ground (**GND**). The circuit is shown in Figure 3. In this setup, **DIO 0** and **DIO 1** are configured as digital output lines and generate a HIGH voltage level, which is used as a power source for the two button branches. For each branch, the upper node is connected to a digital output line (**DIO 0** for **BTN1** and **DIO 1** for **BTN2**), while the junction between the button and resistor is connected to a digital input line (**DIO 2** for **BTN1** and **DIO 3** for **BTN2**). The resistors provide a defined LOW state by pulling the input nodes to ground when the buttons are not pressed. When a button is pressed, the HIGH voltage from the corresponding digital output is applied to the associated input node, causing its state to change from LOW to HIGH. This transition is detected by the device through its digital input channels, allowing real-time monitoring of button presses. This setup provides a practical demonstration of how mechanical button actions can be converted into stable digital signals for measurement and control applications.

Important! Refer to **Table 1** to determine the exact physical pin, connector, or cable that corresponds to each generic connection on the specific Digilent Analog Discovery device being used. The remaining rows in the table show how each generic channel maps the actual hardware connections for each device.

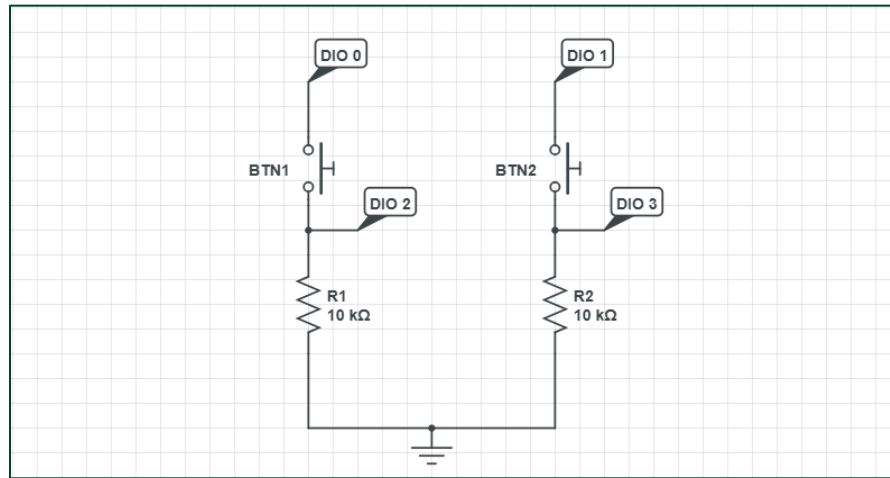


Figure 3. Push-Button Digital Input Circuit with Pull-Down Resistors for Device

4.2 Step by step instructions³

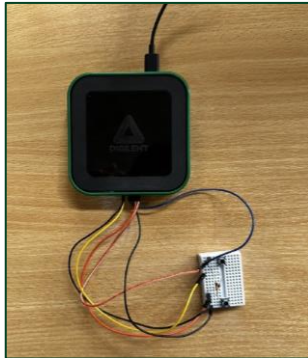
1. Place the first push button (**BTN1**) on the Canvas breadboard, ensuring its two terminals are inserted into different rows.
2. Place the second push button (**BTN2**) on the breadboard in a similar way, ensuring proper separation of terminals.
3. Place the first 10 kΩ resistor (**R1**) on the breadboard and connect one terminal to **GND**.
4. Place the second 10 kΩ resistor (**R2**) on the breadboard and connect one terminal to **GND**.
5. Connect the lower terminal of **BTN1** to the upper terminal of **R1**. This junction will serve as the measurement node.
6. Connect the lower terminal of **BTN2** to the upper terminal of **R2**. This junction will serve as the second measurement node.
7. Connect the junction of **BTN1** and **R1** to **DIO 2**.
8. Connect the junction of **BTN2** and **R2** to **DIO 3**.
9. Connect the upper terminal of **BTN1** to **DIO 0**.
10. Connect the upper terminal of **BTN2** to **DIO 1**.
11. Ensure that all ground connections are common and properly connected to the **Device GND**.

Important! Position the thermistor close to the lamp (but not touching it) so the lamp heat changes the thermistor temperature gradually and safely.

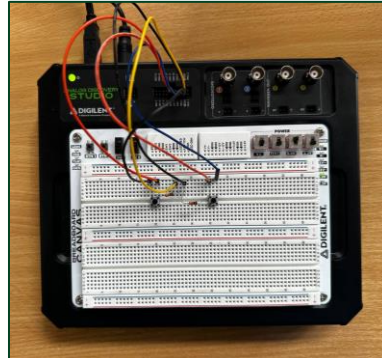
³ Given on Figure 3.

4.3 Assembled Circuit

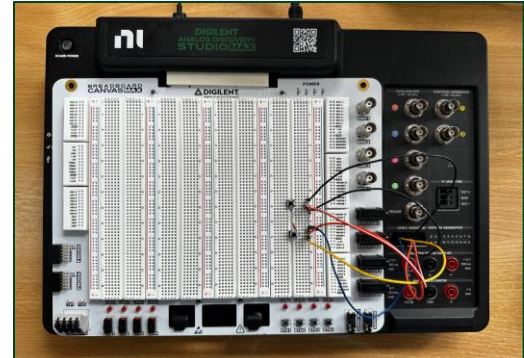
Figure 4 presents complete experimental circuits on different Digilent devices.



a. Analog Discovery 3



b. Analog Discovery Studio



c. Analog Discovery Studio MAX

Figure 4. Assembled circuits for different Devices

5. Software Setup - step by step instructions

5.1 Software Setup in Waveforms

Step 1. Launch WaveForms and connect the device

- 1.1. Connect the **Device** to the computer via USB.
- 1.2. Open the **Digilent WaveForms** software.
- 1.3. Verify that the device is detected and shown as *Currently Selected* in the **Device Manager** (Settings -> Device Manager).

Step 2. Configure the digital outputs in StaticIO

- 2.1. Open the **StaticIO** instrument in **WaveForms**.
- 2.2. In the opened window, locate **DIO 0** and **DIO 1**.
- 2.3. For both **DIO 0** and **DIO 1**, set the output mode to **Push/Pull (1/0)**.
- 2.4. Change the output value of **DIO 0** and **DIO 1** to **1**.

This configuration sets a HIGH voltage level on pins DIO 0 and DIO 1, which are used as power sources for the two button circuits.

Figure 5 shows the configuration of the **Device** power supply using the Digilent WaveForms environment.

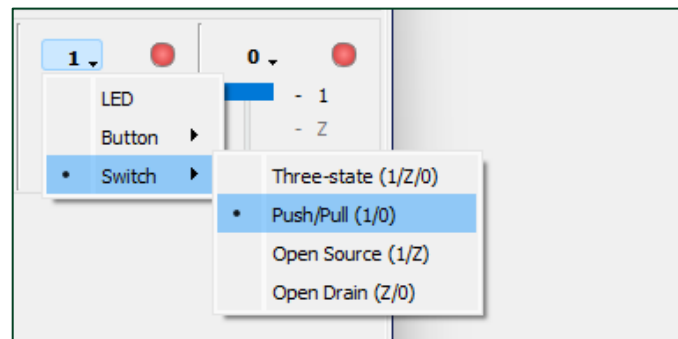


Figure 5. DIO 0 & 1 Configuration in WaveForms Software

Step 3. Observe button states using digital inputs

- 3.1. In the same **StaticIO** window, observe the indicators corresponding to **DIO 2** and **DIO 3**.
- 3.2. Press **BTN1** and verify that the indicator for **DIO 2** changes state.
- 3.3. Release **BTN1** and verify that the indicator for **DIO 2** returns to its initial state.
- 3.4. Repeat the same procedure for **BTN2** and verify the state change of **DIO 3**.

When a button is pressed, the corresponding digital input receives the HIGH voltage from the associated digital output, causing the indicator LED in WaveForms to turn on. When the button is released, the pull-down resistor returns the input to LOW, and the indicator turns off.

Figure 6 shows the real-time button state monitoring in the WaveForms **StaticIO** instrument. In the illustrated example, the indicator for **DIO 3** is active, showing that the corresponding button is pressed. This demonstrates how the digital input changes from LOW to HIGH when the button connects the output voltage to the measurement node.

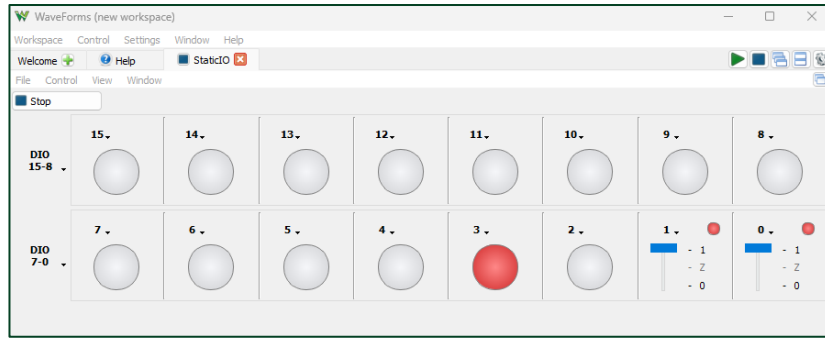


Figure 6. Button Output Signals Displayed in WaveForms

5.2 Data Visualization Using NI LabVIEW WaveForms Toolkit

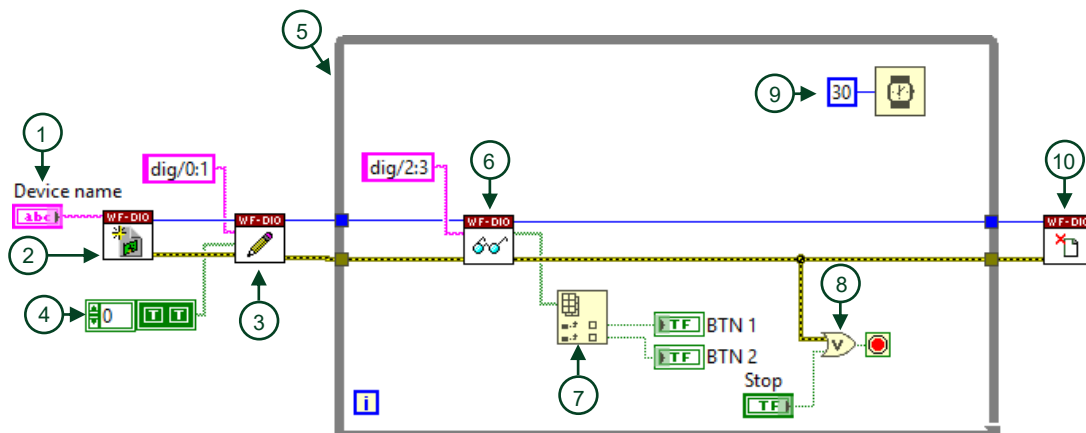
Step 1. Launch NI LabVIEW and connect the Device

- 1.1. Connect the **Device** to the computer via USB.
- 1.2. Open **Diligent WaveForms** and verify that **Device** is detected and shown as *Currently Selected* in the **Device Manager** (Settings -> Device Manager), then close **Diligent WaveForms** software⁴.
- 1.3. Open the **NI LabVIEW** software.
- 1.4. Create a new NI LabVIEW VI (**File->New VI**)

Step 2. Create Block Diagram and Front Panel in NI LabVIEW

- 2.1. Modify the block diagram as shown in **Figure 7**.

Figure 7 VI Block Diagram



1. **Device Name** - specify the name of your **Device**. Use **Table 2** for your **Device**.
2. **WF-Digital Initialize** - this VI initializes the scope.
3. **WF-Digital Write** - specify the digital output channels used in the project. In this case, **dig/0:1** is used to set **DIO 0** and **DIO 1** as outputs that provide HIGH voltage for the button circuits.
4. **Boolean Array** - defines the digital output values written to the selected channels. Here, both values are set to **TRUE**, so **DIO 0** and **DIO 1** output a **HIGH** logic level.

⁴ Before running NI LabVIEW, please ensure that the Diligent WaveForms software is closed.

5. **While Loop** - used to enable continuous data acquisition.
6. **WF-Digital Read** - reads the states of the digital input channels. In this case, **dig/2:3** is used to read the button signals from **DIO 2** and **DIO 3**.
7. **Index Array** - separates the read digital data array into individual button states so that BTN1 and BTN2 can be processed independently.
8. **OR** - Allow the code to stop using the **Stop** button or automatically when an error occurs.
9. **Wait (ms)** - connected to an integer constant and set to 30 ms to control the loop execution rate and reduce CPU usage.
10. **WF-Digital Close** - closes the digital session, releases the device resources, and resets all values.

- 2.2. Modify the front panel shown in **Figure 8**. Inside **Device name** String Control write your corresponding **Device** name from **Table 2**.

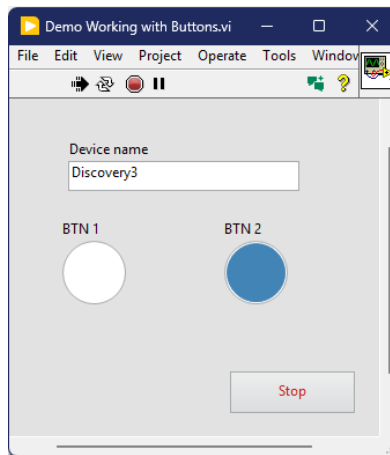


Figure 8. VI Front Panel

- 2.3. Save the VI.

Step 3. Run the code and observe measurements in LabVIEW

- 3.1. Run the code by clicking on the **Run button**.
- 3.2. Press **BTN1** and **BTN2** and observe the corresponding changes in the front panel indicators in real time.

6. Clicker Racing Game in LabVIEW

This case study includes an interactive application called the **Clicker Racing Game**, developed in NI LabVIEW. The purpose of this demonstration is to show how physical push buttons can be integrated with software to create real-time user interaction using digital input signals.

6.1 Description

The Clicker Racing Game uses the same button circuit and digital I/O configuration described earlier. The digital input signals from **DIO 2** and **DIO 3** are read in LabVIEW and used to control game logic. Each button press is interpreted as an action (e.g., advancing a player), demonstrating how hardware inputs can be directly mapped to software behavior.

This example highlights how simple digital circuits can be combined with LabVIEW programming to create interactive applications, bridging the gap between hardware and software systems.

6.2 How to open and run the project:

Step 1. Open the **Clicker Racing Game** folder.

Step 2. Open the **Clicker Racing Game** LabVIEW Project file.

Step 3. In the **Project Explorer**, open **Main.vi**.

Step 4. In the **Device Name** control, specify the name of your **Device**. Use **Table 2** for your **Device**.

Step 5. Ensure the device is properly connected and recognized.

Step 6. Run the VI by clicking the **Run** button.

Step 7. Press **BTN1** and **BTN2** on the circuit.

Step 8. Observe how each button press affects the game interface in real time.

The Front Panel of the Clicker Racing Game in NI LabVIEW is shown in Figure 9, illustrating the user interface used to monitor button inputs and interact with the game in real time.

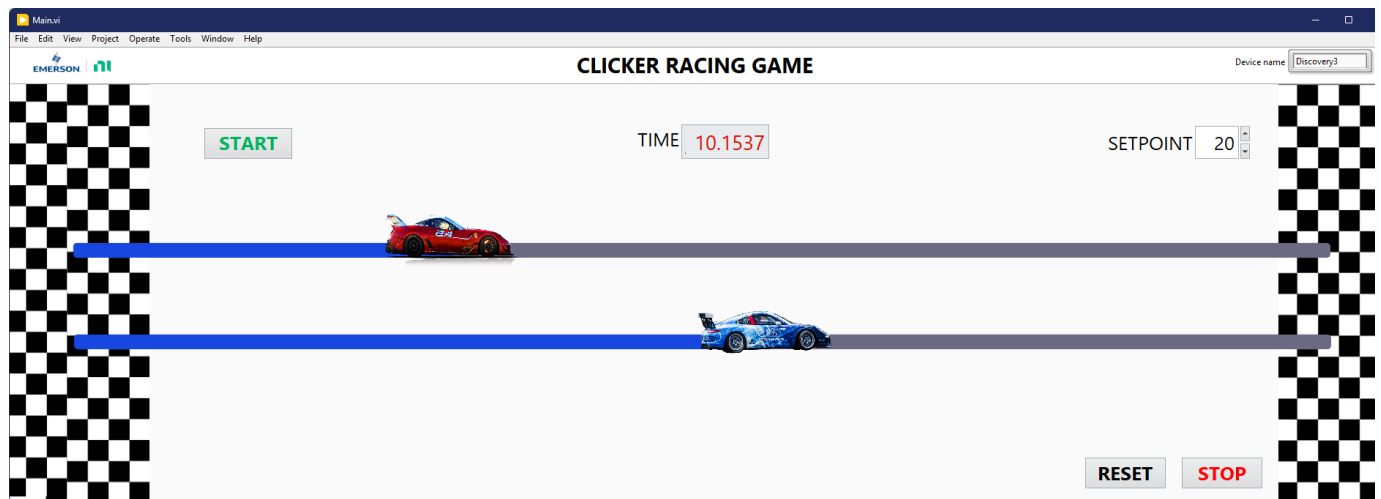


Figure 9. The Front Panel of the Clicker Racing Game developed in NI LabVIEW