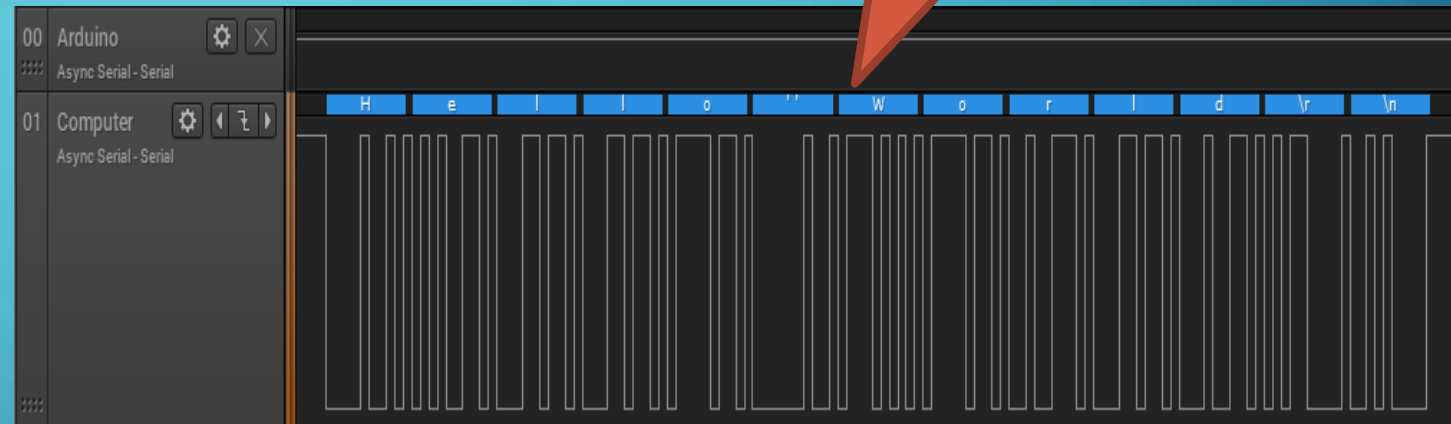


COMMUNICATION PROTOCOLS REVERSE ENGINEERING

LECTURE 1



BY: OMAR MEKKAWY

MY LINKEDIN: [HTTPS://WWW.LINKEDIN.COM/IN/OMAR-MEKKAWY/](https://www.linkedin.com/in/omar-mekrawy/)

GITHUB: [HTTPS://GITHUB.COM/RXTXINV/COMMUNICATION_PROTOCOLS_REVERSE_ENGINEERING_COURSE/](https://github.com/RXTXINV/COMMUNICATION_PROTOCOLS_REVERSE_ENGINEERING_COURSE/)

MY WEBSITE: [HTTPS://OMAR-MEKKAWY.NET](https://omar-mekrawy.net) – [HTTPS://OMAR-MEKKAWY.COM](https://omar-mekrawy.com)

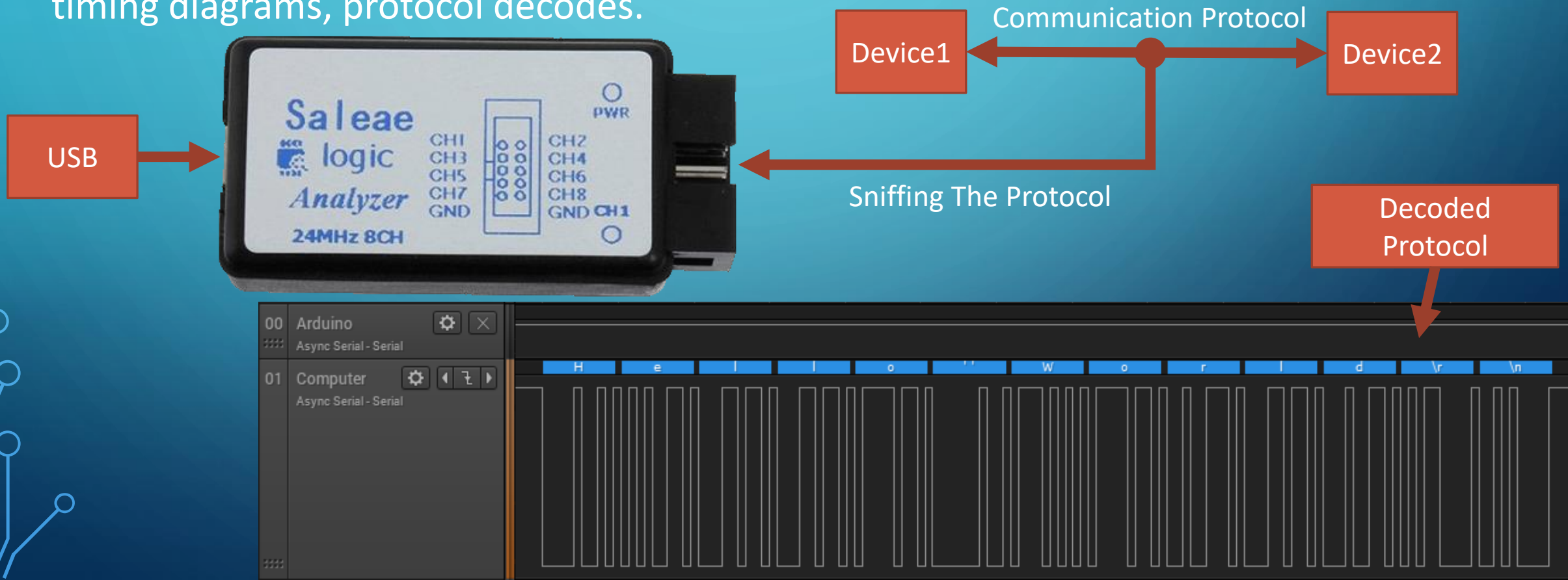
FACEBOOK: [HTTPS://WWW.FACEBOOK.COM/OMARMEKKAWYOFFICIAL/](https://www.facebook.com/OMARMEKKAWYOFFICIAL/)

LECTURE 1 AGENDA

- What is the Logic Analyzer ?
- What is the importance of the logic analyzer ?
- The physical components of the logic analyzer.
- The Software interface.
- Why using Arduino in this course ?
- Example 1 (LED Blink)
- Example for Sniffing UART Protocol.

WHAT IS THE LOGIC ANALYZER

- The logic analyzer is an electronic instrument that captures (Samples) and displays multiple signals from a digital system or digital circuit without affecting the communication between them. A logic analyzer may convert the captured data into timing diagrams, protocol decodes.



WHAT IS THE IMPORTANCE OF THE LOGIC ANALYZER

- Its importance:
 - Useful tool when **learning the communication protocols** (UART, I²C, SPI, 1-Wire, CAN, LIN, SMBus, I²S).
 - Useful tool when debugging the embedded hardware.
 - Useful tool when trying to reverse-engineer the communication protocols for a product.
 - Capturing the data for later use and documentation purposes.
- Its considered as a **skill** for most of the embedded systems companies.
- Supported Protocols:
 - I²C, SPI, 1-Wire, CAN, I²S – PCM, JTAG, LIN, Modbus, SMBus, ... more.
- Support 8 Digital Channels with sampling rate up to 24Ms/S.

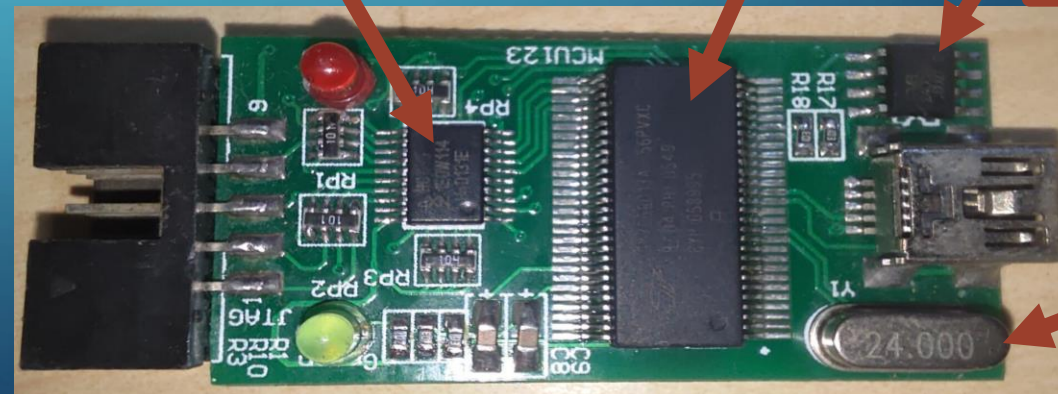
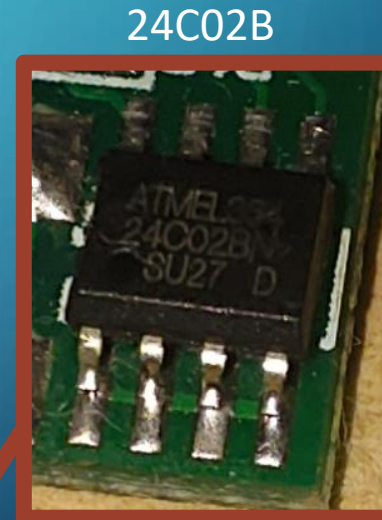
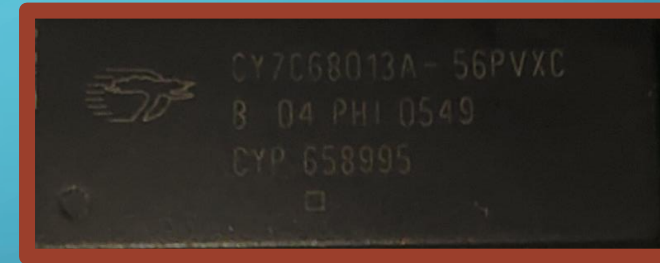
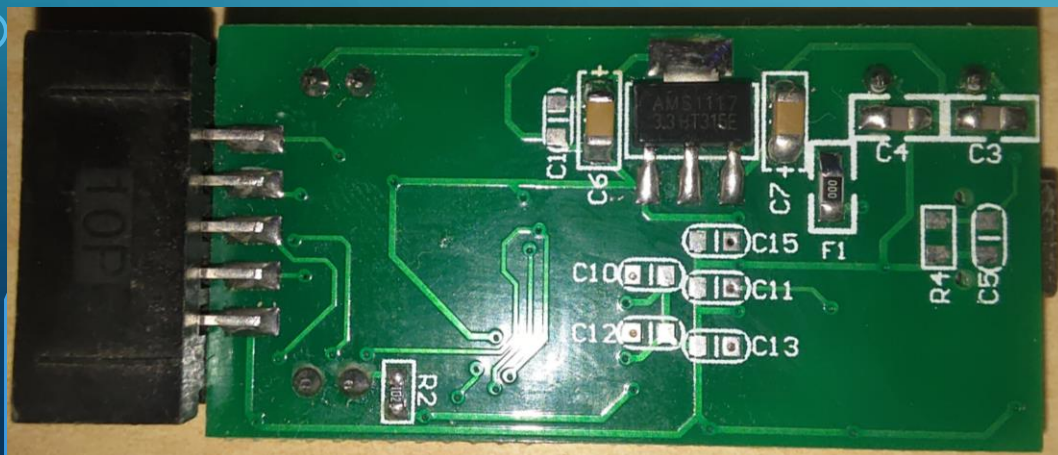
Supported Protocols



- Async Serial
- I2C
- SPI
- Hide
- 1-Wire
- Atmel SWI
- BiSS C
- CAN
- DMX-512
- HD44780
- HDLC
- HDMI CEC
- I2S / PCM
- JTAG
- LIN
- MDIO
- Manchester
- Midi
- Modbus
- PS/2 Keyboard/Mouse
- SMBus
- SWD
- Simple Parallel
- UNI/O
- USB LS and FS

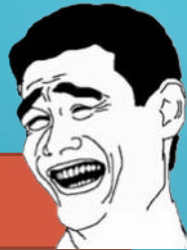
THE PHYSICAL COMPONENTS OF THE LOGIC ANALYZER

- Components of the **Saleae Logic 8 (Chinese Clone)**
 - Cypress CY7C68013A Microcontroller.
 - 74HC245 (Octal Bus Transceiver) used for **level shifting** and **protection**.
 - 24C02B EEPROM.

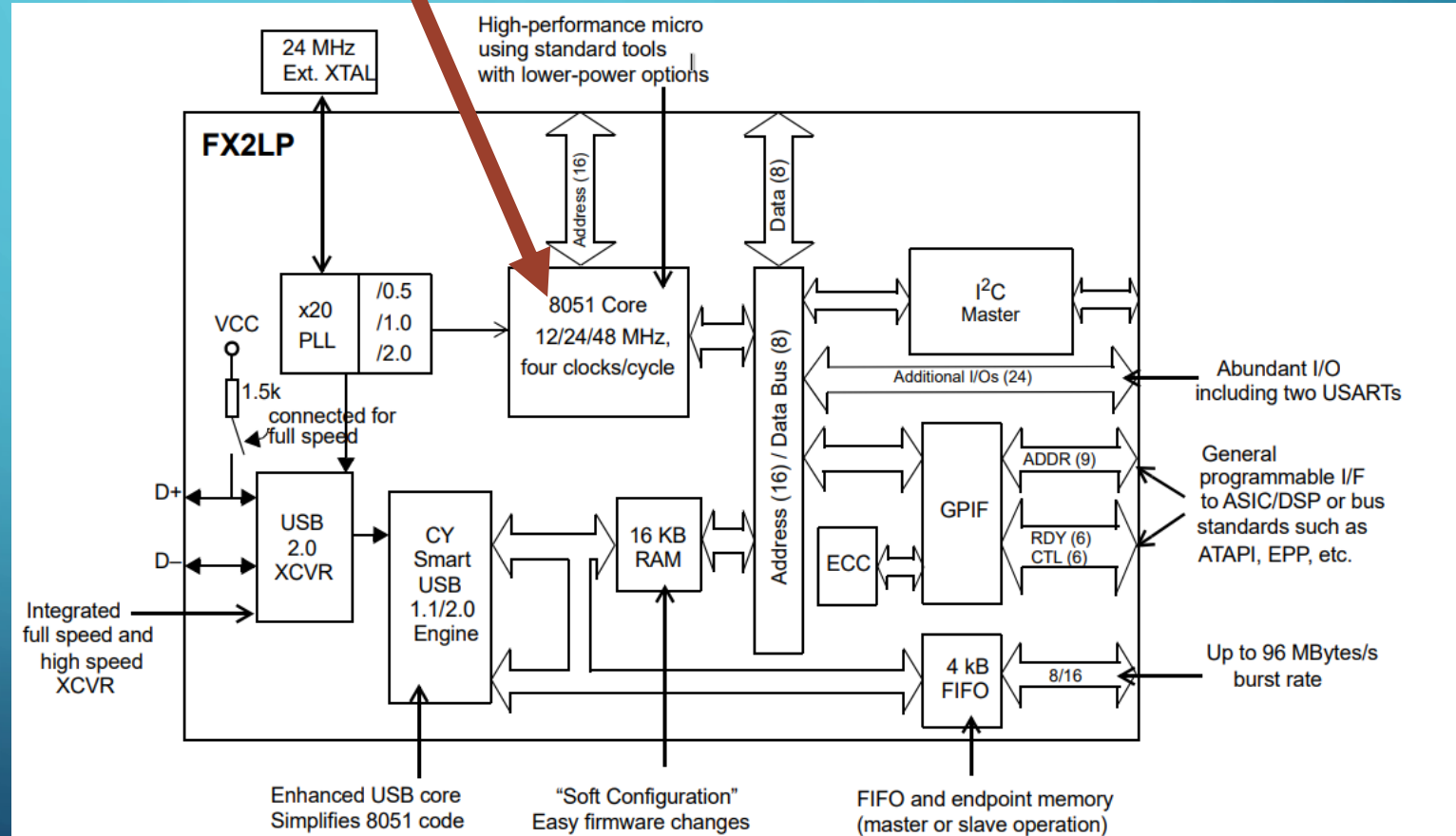


24MHz
Crystal

Cypress CY7C68013A



Confirmed, **8051** Is Here !!



THE PHYSICAL COMPONENTS OF THE LOGIC ANALYZER

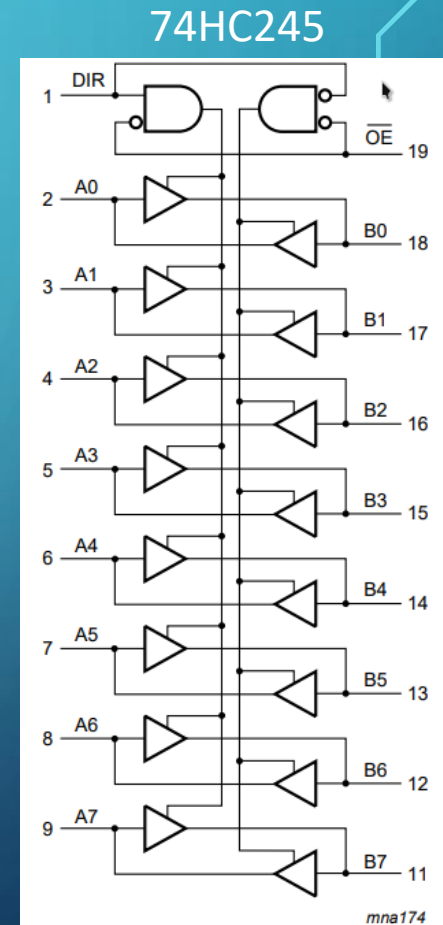
74HC245



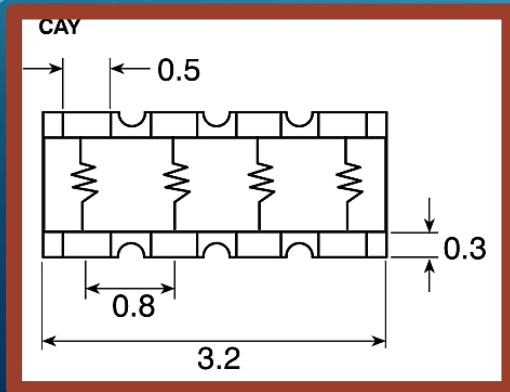
Maximum Input Voltage = 5V



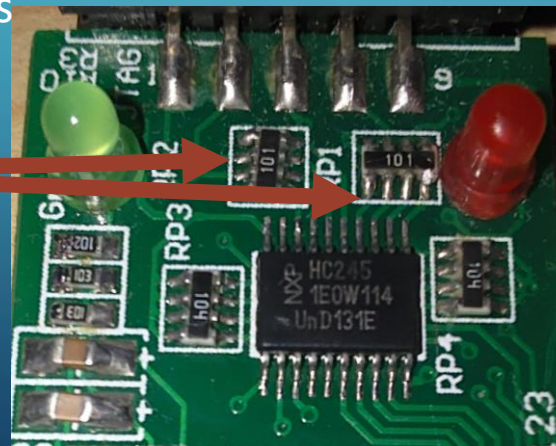
The 74HC245; 74HCT245 is an 8-bit transceiver with 3-state outputs. The device features an output enable (\overline{OE}) and send/receive (DIR) for direction control. A HIGH on \overline{OE} causes the outputs to assume a high-impedance OFF-state. Inputs include clamp diodes. This enables the use of current limiting resistors to interface inputs to voltages in excess of V_{CC} .



Current Limiting Resistors



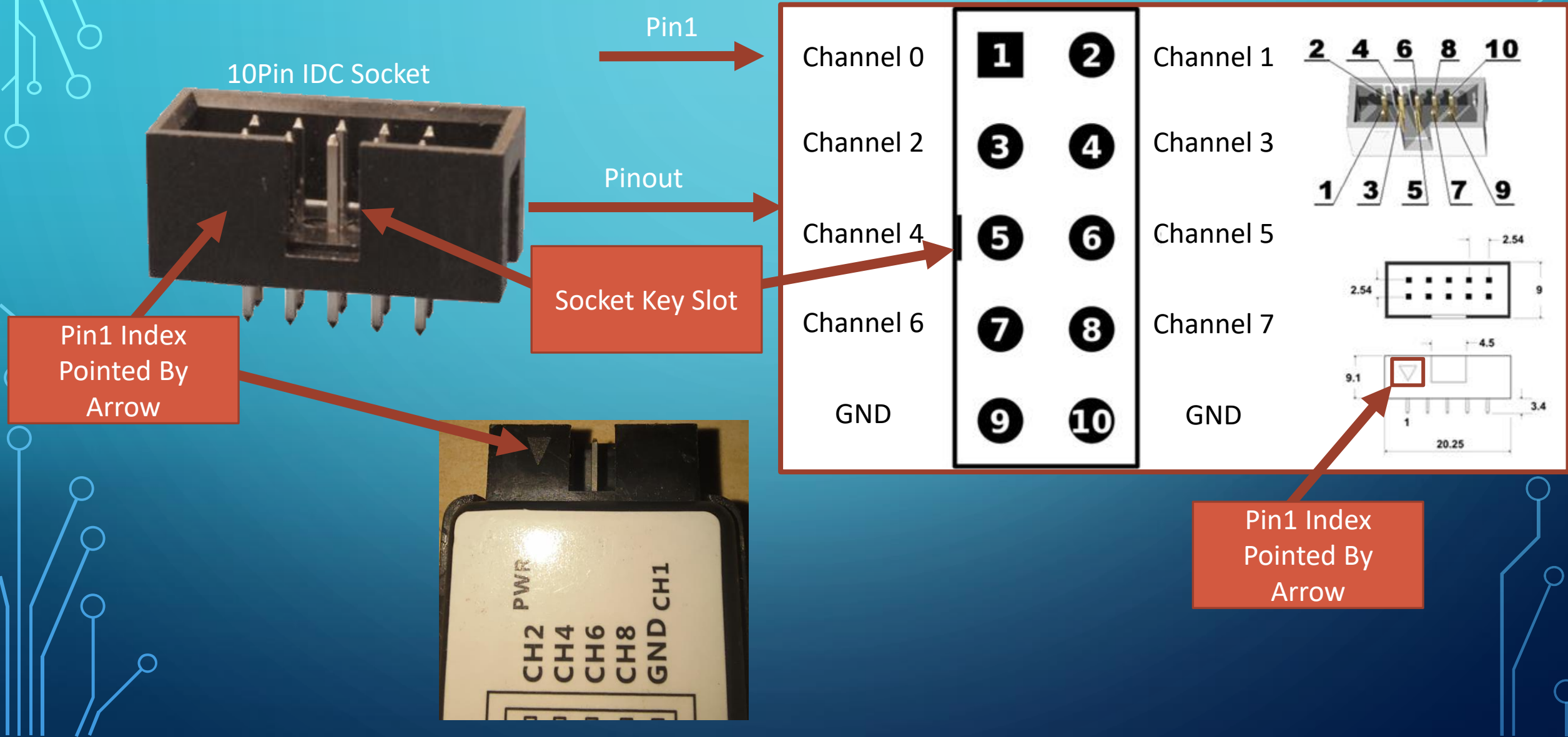
100 Ohm
Resistor
Network



Power
LED

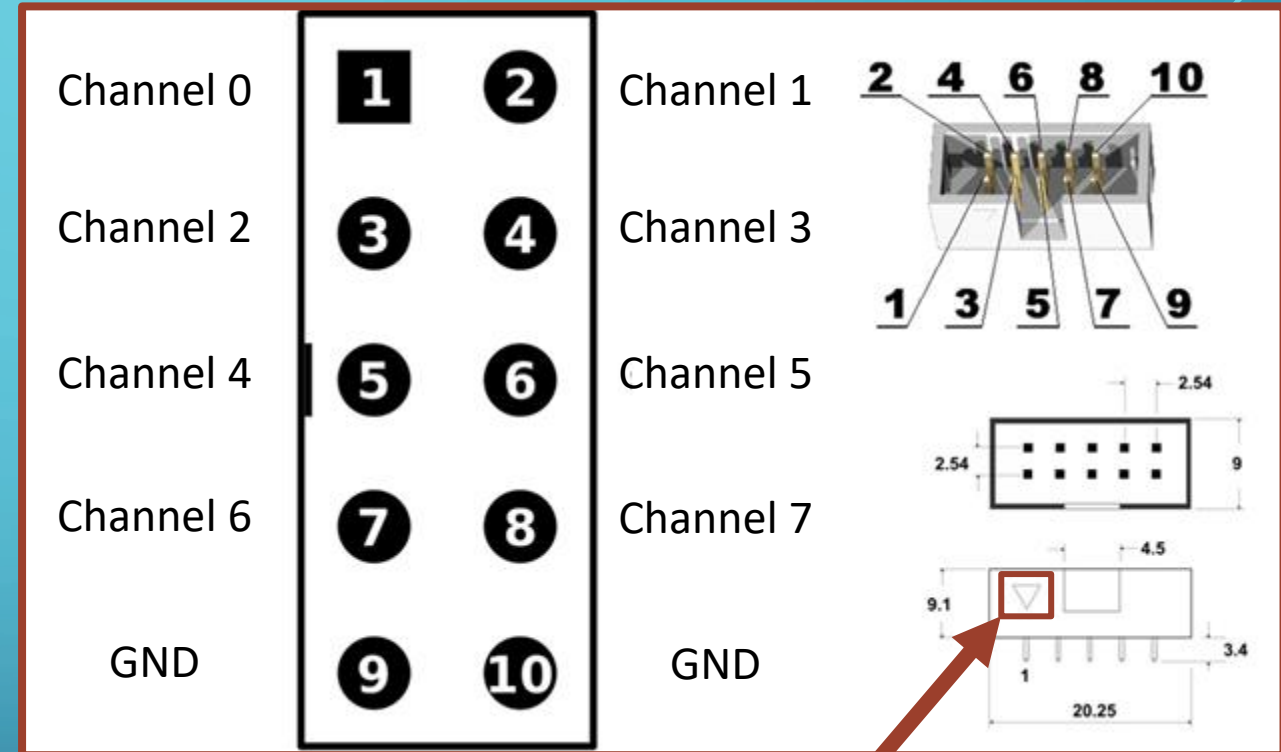
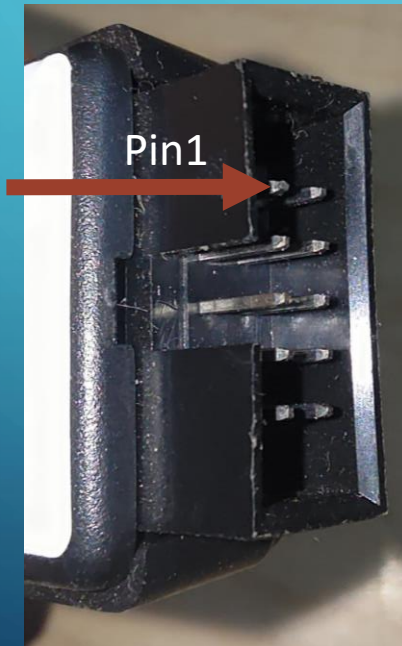
Channel 0
LED

THE PHYSICAL COMPONENTS OF THE LOGIC ANALYZER



THE PHYSICAL COMPONENTS OF THE LOGIC ANALYZER

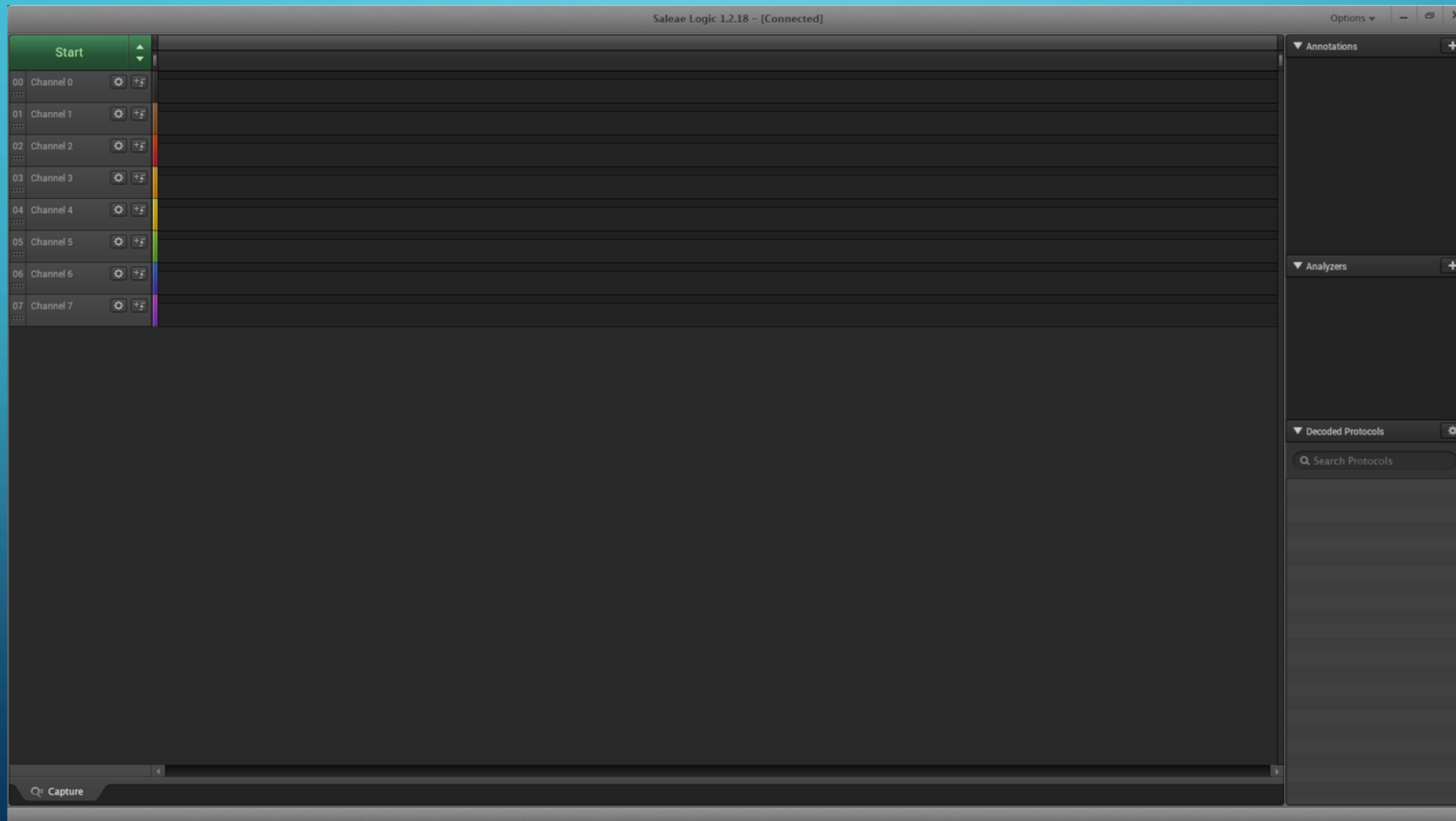
Pin1 Index
Pointed By
Arrow



Pin1 Index
Pointed By
Arrow

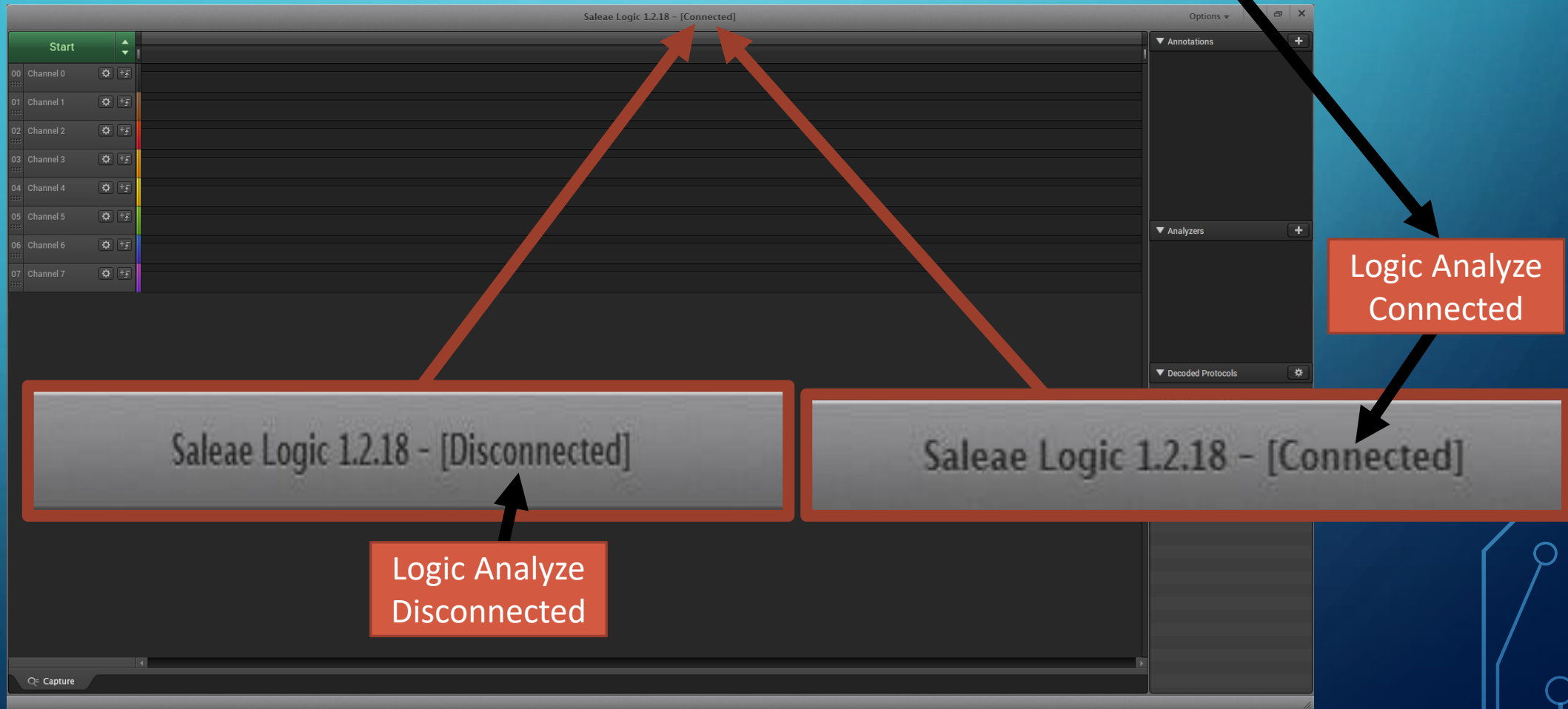
THE SOFTWARE INTERFACE

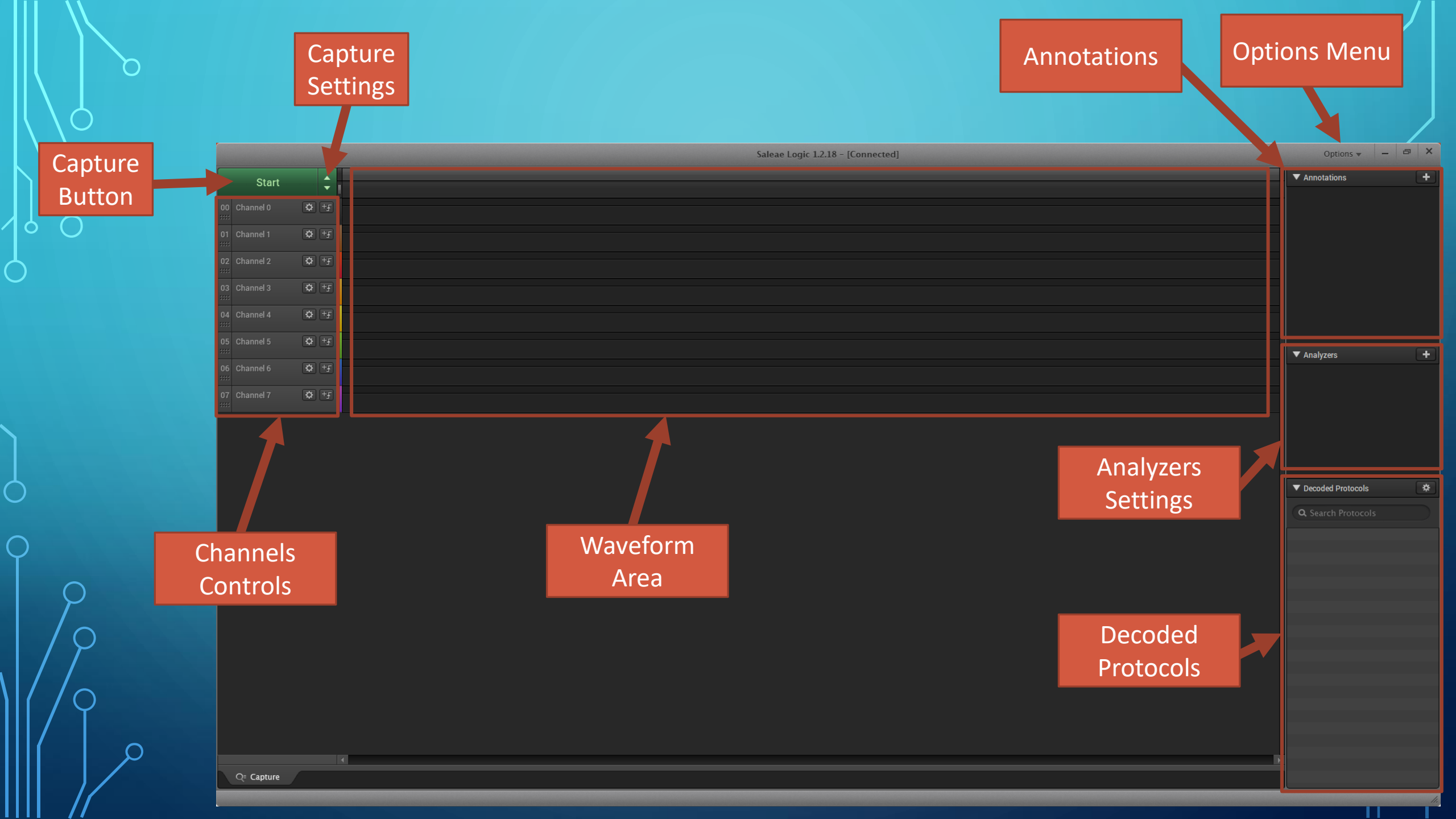
- Software Installing & How To Use:
 - Download the software [Saleae Logic 1.2.18](#) and install it like any program.



THE SOFTWARE INTERFACE

USB





Capture Button

Capture Settings

Annotations

Options Menu

Channels Controls

Waveform Area

Analyzers Settings

Decoded Protocols

ANNOTATIONS

- Used for:

- Adding Bookmarks, timing markers, and various measurements

Press +

Annotations

Bookmark
Timing Marker Pair
Measurement
Remove all annotations

Annotations

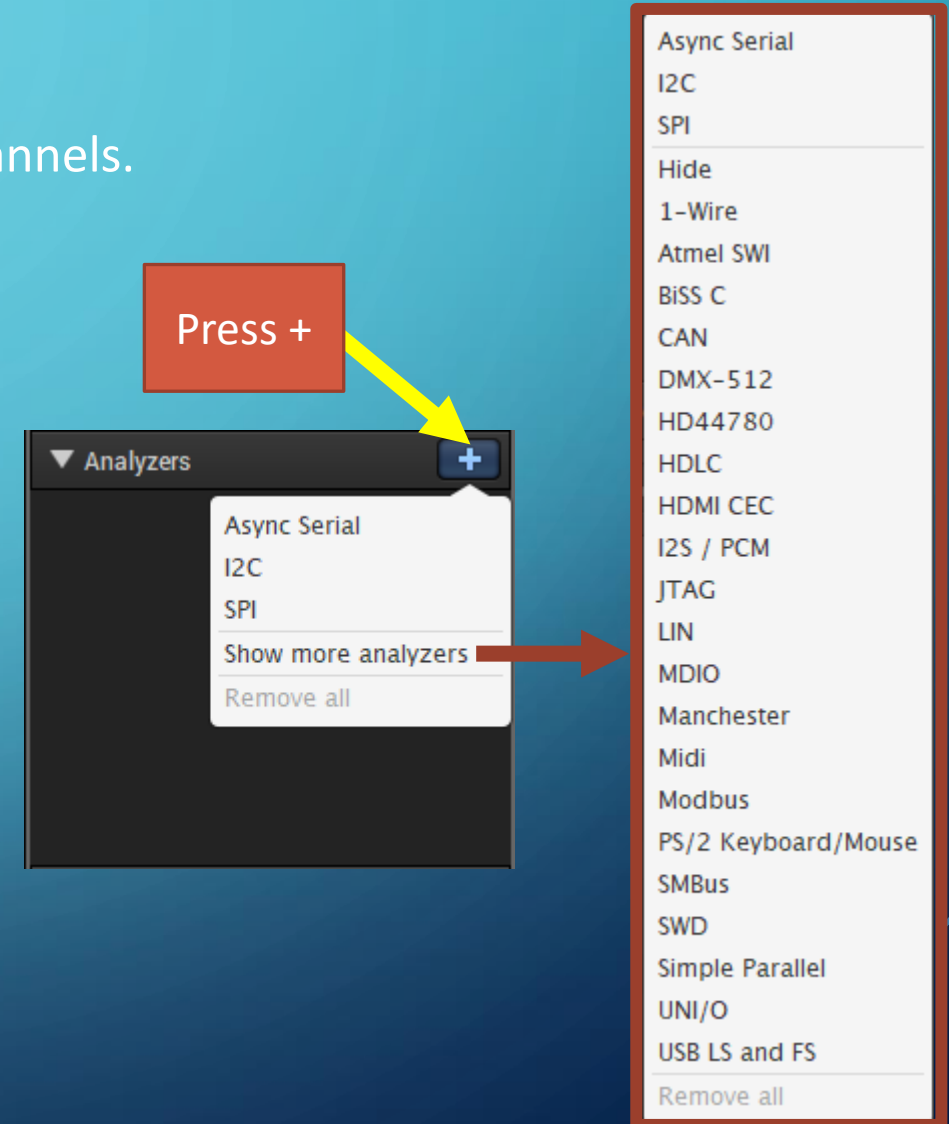
Timing Marker Pair

| A1 - A2 | = 48.29166667 μ s
A1 @ 16.03433617 s
A2 @ 16.03438446 s

Measuring Time

ANALYZERS

- Used for:
 - Adding one/many protocol analyzer(s) to the digital channels.

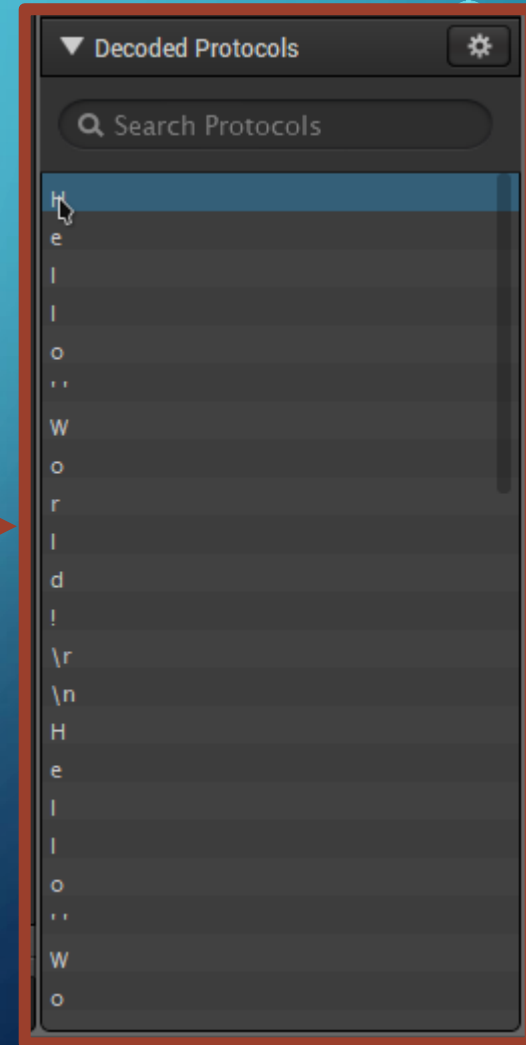
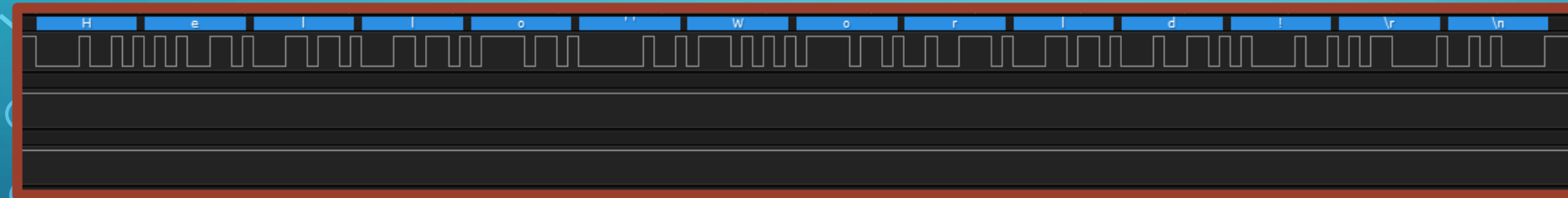


DECODED PROTOCOLS

- Used for:

- Showing the decoded results of the protocol analyzers once the data has been processed.
- Search within the decoded data.

Decoded UART Data



CHANNELS CONTROLS

Decoded UART Data

- Used for:

- Controlling the channels (Trigger, Zoom, Sort channels, Hide Channels)
- Controlling the data capture (Sampling Rate, Sampling Duration)

The image shows a software interface for managing channels. On the left, a list of 8 channels (00 to 07) is displayed, each with a gear icon and a '+f' icon. A red box highlights this list. An arrow points from this list to a larger, detailed view of Channel 5 settings on the right. A yellow arrow points to the gear icon for Channel 5 in the detailed view, with a red box labeled 'Click on' and a gear icon. The 'Channel Settings' menu for Channel 5 includes zoom options (1x, 2x, 4x, 8x), a 'Show All Channels' button, a 'Hide This Channel' button, a 'Reset All Channels' button, and an 'Enable Glitch Filter' button. Red boxes with arrows point to each of these menu items, with labels: 'Zoom (1,2,4,8)x', 'Hide Channel', 'Reset All Channels', and 'Enable Glitch Filter'.

Start

00 Channel 0

01 Channel 1

02 Channel 2

03 Channel 3

04 Channel 4

05 Channel 5

06 Channel 6

07 Channel 7

Click on

Channel Settings

1x 2x 4x 8x

Show All Channels

Hide This Channel

Reset All Channels

Enable Glitch Filter

Zoom (1,2,4,8)x

Hide Channel

Reset All Channels

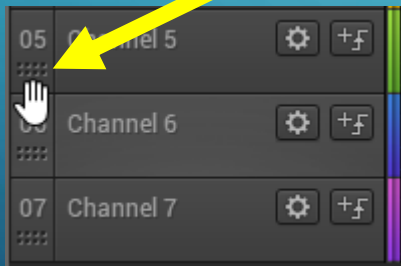
Enable Glitch Filter

CHANNELS CONTROLS

- Used for:

- Controlling the channels (Trigger, Zoom, Sort channels, Hide Channels)
- Controlling the data capture (Sampling Rate, Sampling Duration)

Hover on  and drag (up/down) to move the channel

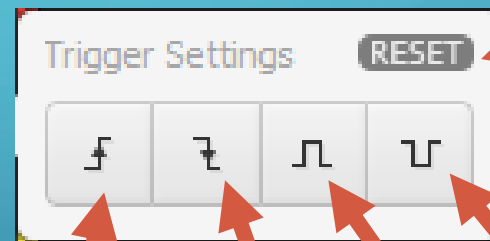
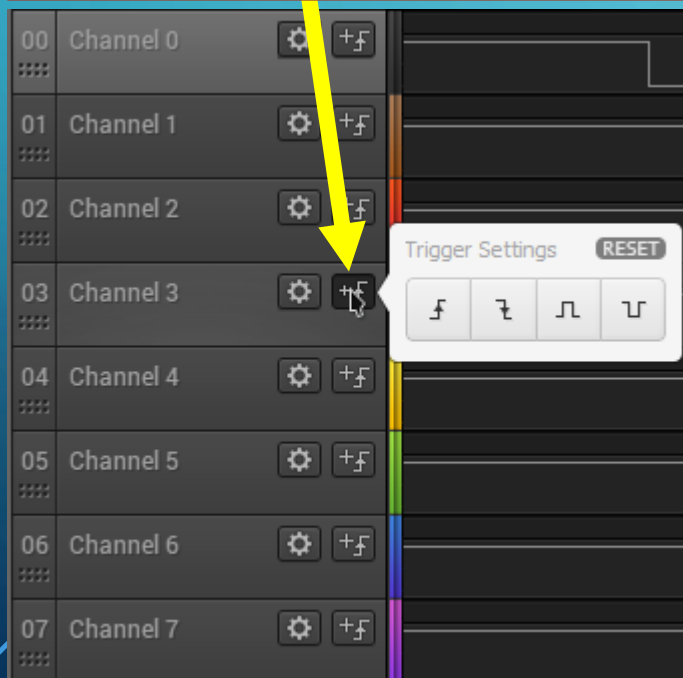


CHANNELS CONTROLS

- Used for:

- Controlling the channels (Trigger, Zoom, Sort channels, Hide Channels)
- Controlling the data capture (Sampling Rate, Sampling Duration)

Click on  to choose triggering settings for the channel



Reset Trigger (Remove Trigger)

Trigger on Rising Edge

Trigger on Falling Edge

Trigger on Negative Pulse Width

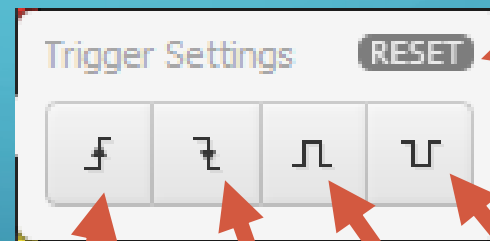
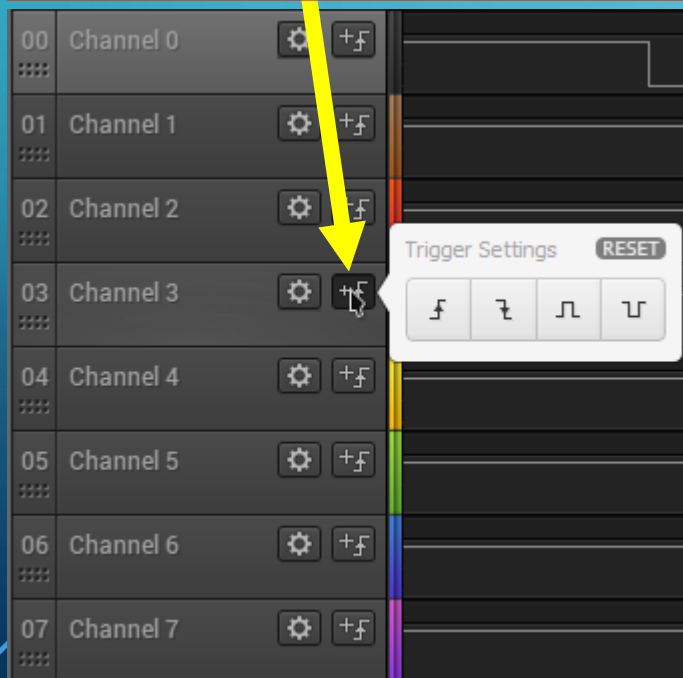
Trigger on Positive Pulse Width

CHANNELS CONTROLS

- Used for:

- Controlling the channels (Trigger, Zoom, Sort channels, Hide Channels)
- Controlling the data capture (Sampling Rate, Sampling Duration)

Click on  to choose triggering settings for the channel



Reset Trigger (Remove Trigger)

Trigger on Rising Edge

Trigger on Falling Edge

Trigger on Negative Pulse Width

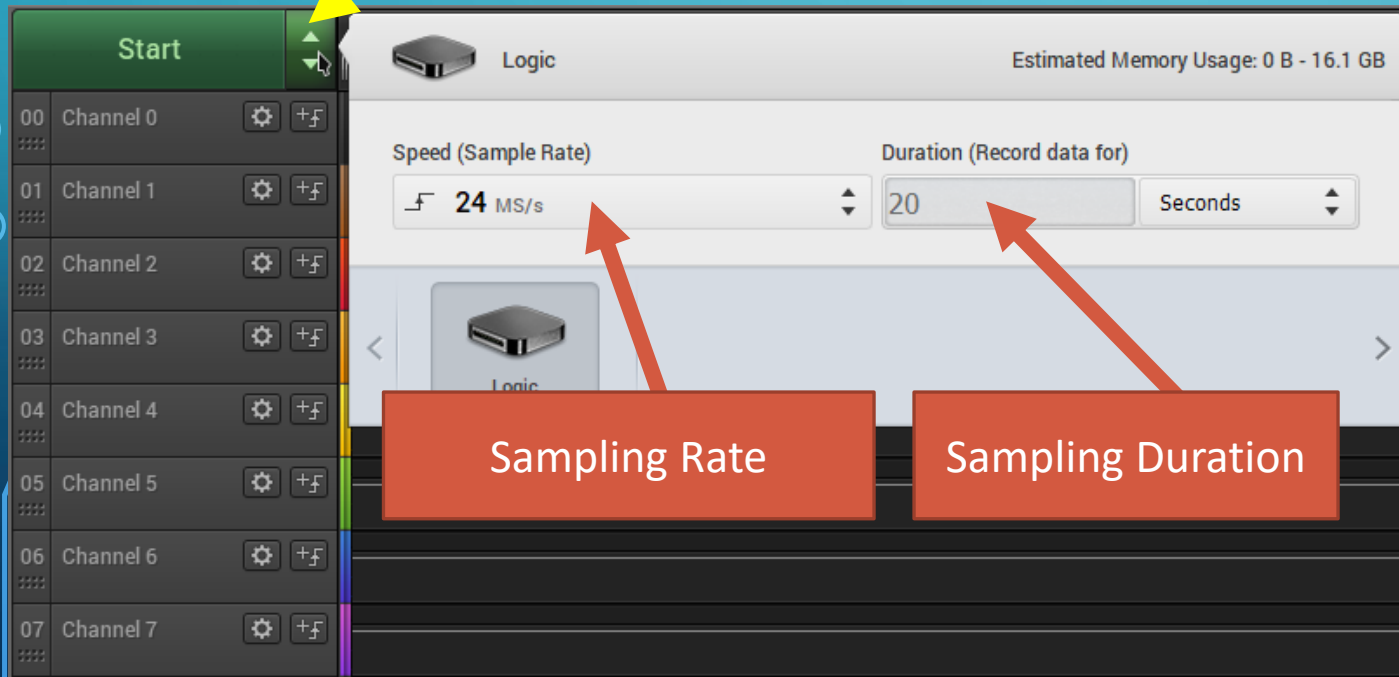
Trigger on Positive Pulse Width

CHANNELS CONTROLS

- Used for:

- Controlling the channels (Trigger, Zoom, Sort channels, Hide Channels)
- Controlling the data capture (Sampling Rate, Sampling Duration)

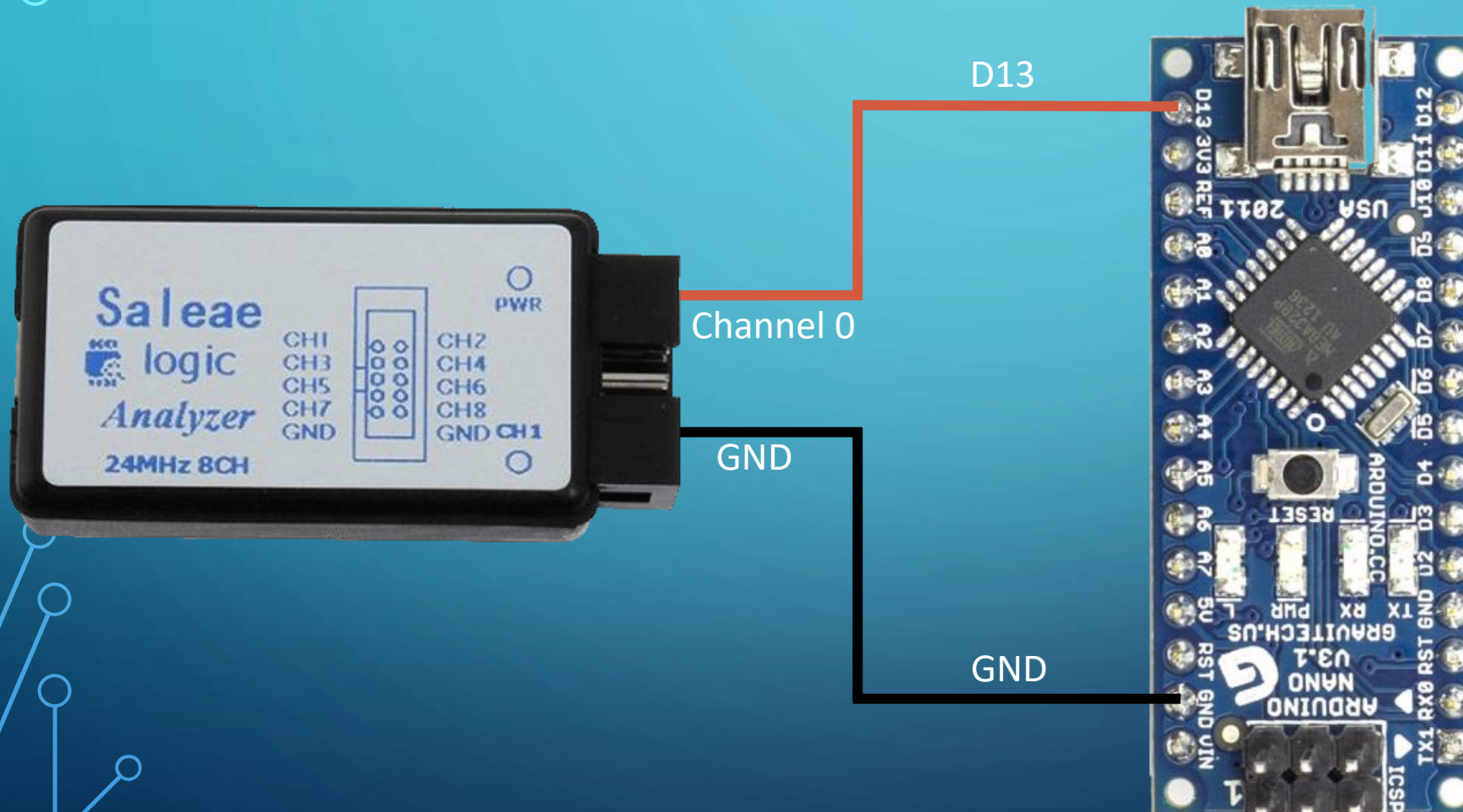
Click on  to choose triggering settings for the channel



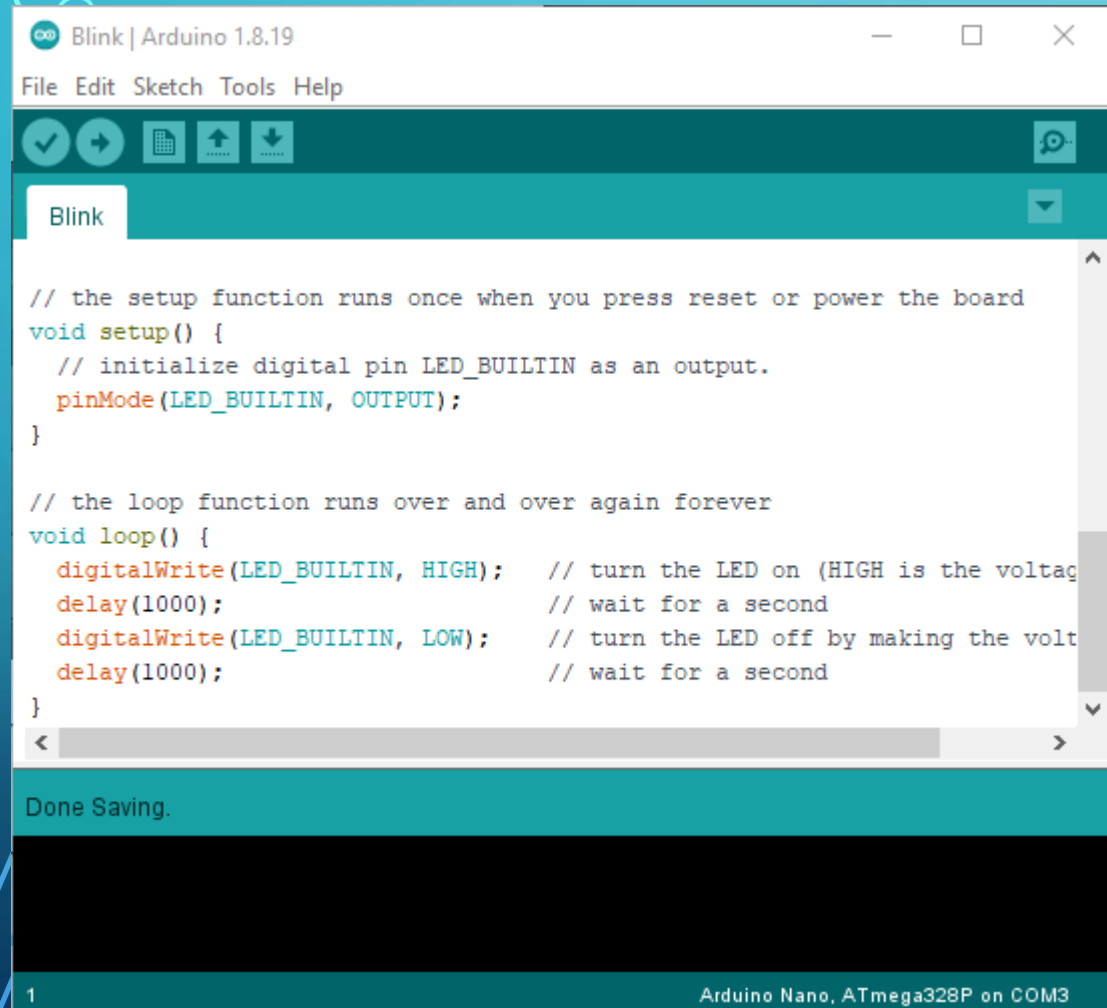
The screenshot displays the Logic Analyzer software interface. On the left, a vertical list of channels (00 to 07) is shown, each with a gear icon and a waveform icon. A yellow arrow points from the 'Start' button to the trigger icon in the channel list. The main window shows the 'Logic' tab with 'Estimated Memory Usage: 0 B - 16.1 GB'. Below this, the 'Speed (Sample Rate)' is set to '24 MS/s' and the 'Duration (Record data for)' is set to '20 Seconds'. Two red arrows point from labels 'Sampling Rate' and 'Sampling Duration' to their respective fields. The channel list on the left includes: Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, and Channel 7.

EXAMPLE 1 (LED BLINK)

- This example blinks a LED every 1 second, lets record its GPIO data.



EXAMPLE 1 (LED BLINK)

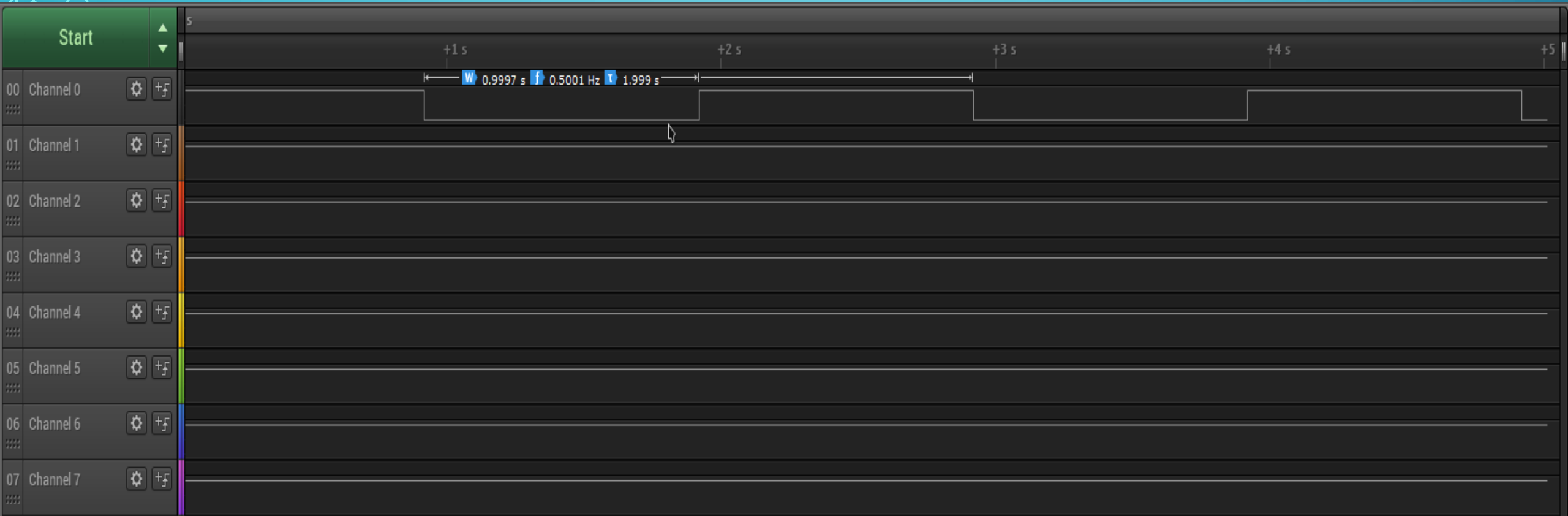
A screenshot of the Arduino IDE interface. The window title is "Blink | Arduino 1.8.19". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". The toolbar shows icons for opening, saving, and running. The sketch name "Blink" is displayed in a tab. The code editor contains the following C++ code:

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);                     // wait for a second
  digitalWrite(LED_BUILTIN, LOW);  // turn the LED off by making the voltage LOW
  delay(1000);                     // wait for a second
}
```

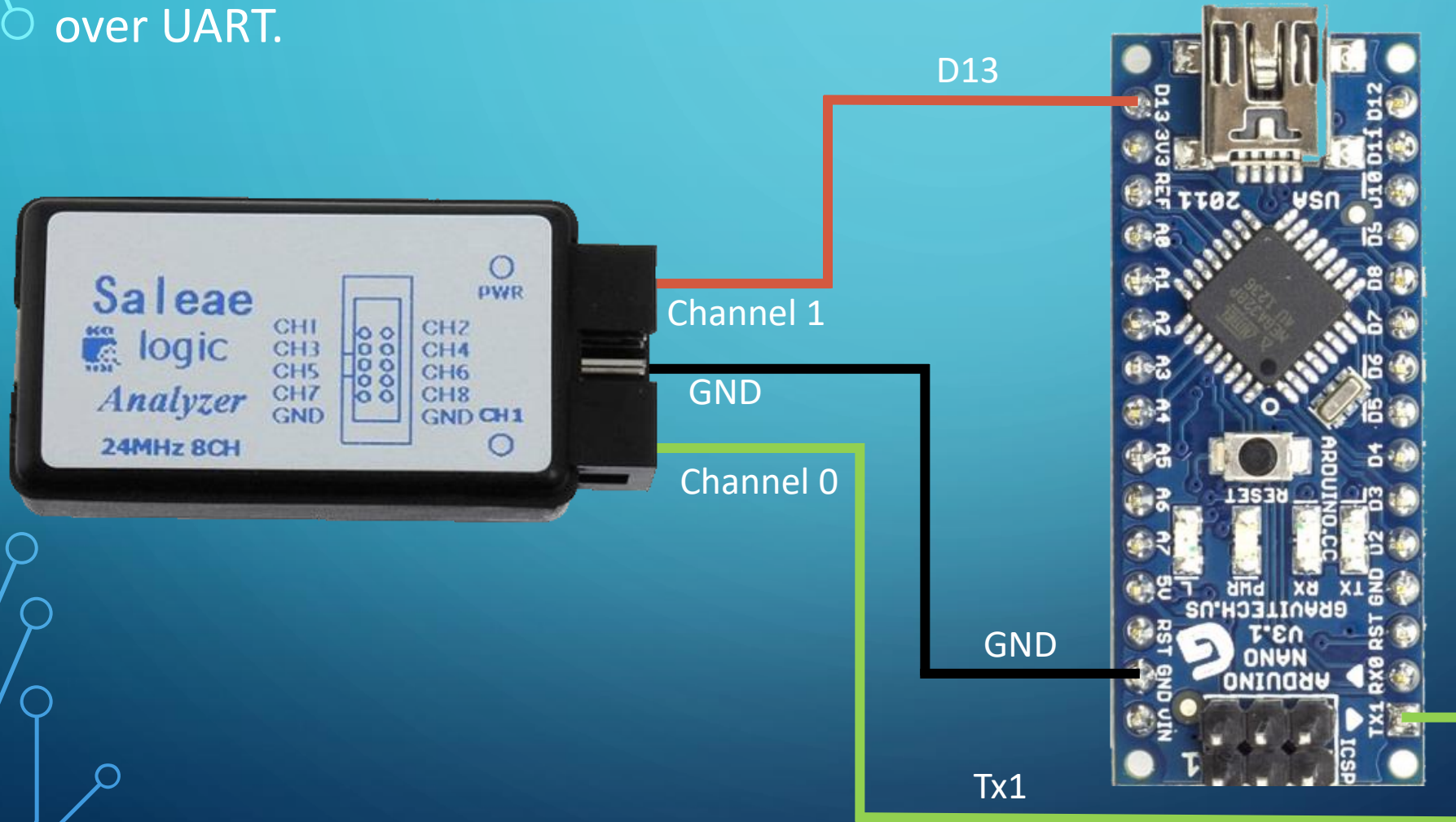
A status bar at the bottom indicates "Done Saving." and "1 Arduino Nano, ATmega328P on COM3".

EXAMPLE 1 (LED BLINK)



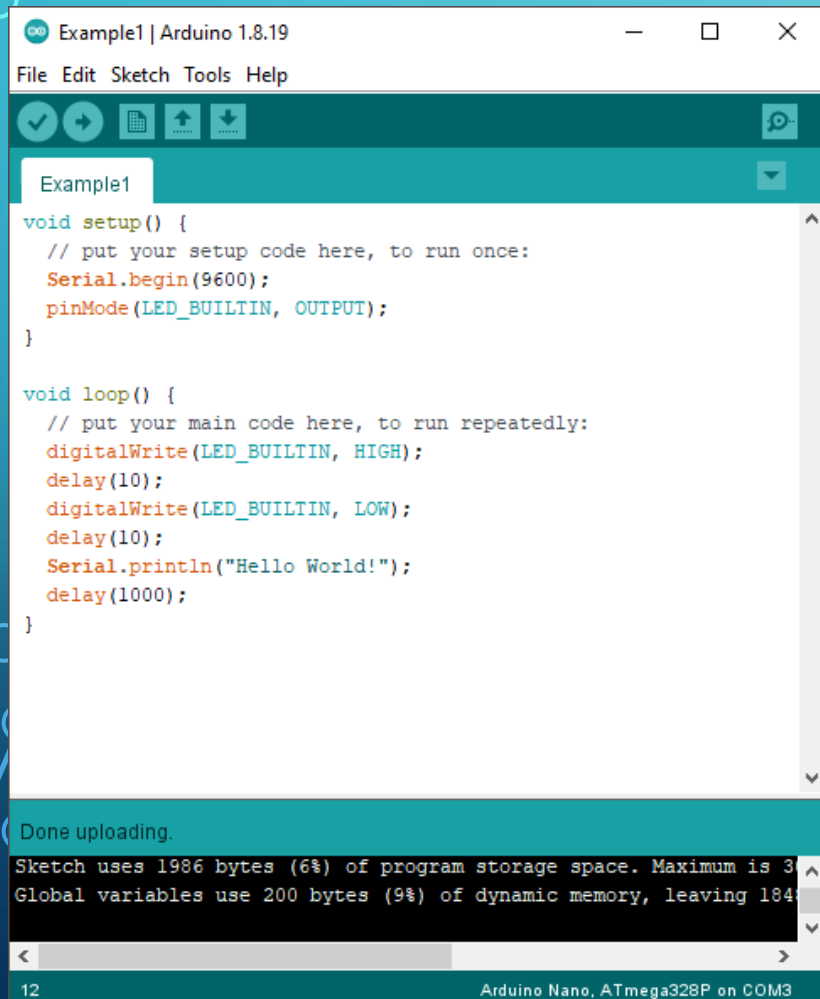
EXAMPLE 2 (LED BLINK + UART)

- This example generates a pulse with duration of 10ms on LED and sends “Hello World” over UART.



EXAMPLE 2 (LED BLINK + UART)

- This example blinks a LED every 1 second, lets record its GPIO data.



The screenshot shows the Arduino IDE interface. The title bar reads "Example1 | Arduino 1.8.19". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". Below the menu bar is a toolbar with icons for checking, running, uploading, and downloading. The main editor area shows the following code:

```
void setup() {  
  // put your setup code here, to run once:  
  Serial.begin(9600);  
  pinMode(LED_BUILTIN, OUTPUT);  
}  
  
void loop() {  
  // put your main code here, to run repeatedly:  
  digitalWrite(LED_BUILTIN, HIGH);  
  delay(10);  
  digitalWrite(LED_BUILTIN, LOW);  
  delay(10);  
  Serial.println("Hello World!");  
  delay(1000);  
}
```

At the bottom, the status bar shows "12" and "Arduino Nano, ATmega328P on COM3". The output window at the bottom displays the following text:

```
Done uploading.  
Sketch uses 1986 bytes (6%) of program storage space. Maximum is 3  
Global variables use 200 bytes (9%) of dynamic memory, leaving 184
```

EXAMPLE 2 (LED BLINK + UART)

- This example blinks a LED every 1 second, lets record its GPIO data.

