Custom VR

This Document provides instructions for creating the custom VR headset described in the previous chapters. It begins by listing the required parts for each assembly step and provides detailed instructions for building the hardware. A description of the compilation process for Monado follows this. Since Monado does not run as expected by simply starting the program, the necessary environment variables are listed. Finally, if game support via SteamVR is desired, this chapter also includes an explanation of the required modifications.

1 Assembling the Headset

To create a new headset, a 3D printer, a Metal Saw, a lighter, or other source of heat, and a soldering iron are required. In Table 1, the parts that need to be purchased are listed. For 3D printing, a total of 39 parts need to be printed, 10 for each eye module, 8 for the case, and 1 for the electronics compartment. The assembly process is divided into two parts. The first describes the construction of the left eye module, which can also be used for the right eye module. For the second part, the full assembly of the case with the eye modules and the electronic compartment is described.

1.1 Eye Module

In Table 2, the required parts for the headset are listed, and in Figure 1, the 3D printed parts are listed and enumerated. The first step is to prepare the threaded rods. Using a metal saw, cut a 3mm deep line in the center of the

PartN	Count	Product ID
Threaded Rod M3 55mm	4	B0D5DGMXNN (Amazon)
Press Insert M3	4	B0CRYWCYMG (Amazon)
Allen Screw M2 4mm	40	B0CKN2VK12 (Amazon)
Press Insert M2	41	B0CRYWCYMG (Amazon)
Allen Screw M2 6mm	1	B0CKN2VK12 (Amazon)
Display with Breackoutboard	2	1005008477815602 (AliExpress)
MIPI-to-HDMI Board	2	1005008477815602 (AliExpress)

Table 1: Components which need to be bought with their corresponding Product ID and vendor

(a) Inner Module (b) Outer Module

Figure 1: 3D printed Parts for the eye module with an enumeration

front side of the rod. This cut divides one end of the rod into two halves. Remove one half so that a 3 mm high semicircle remains, connected to the rest of the threaded rod. This semicircle is used to connect to part 5 and transfer the rotational movement from the gear to the threaded rod. By heating the cut end, the rod can be pressed into the semicircular hole in the center of part 5. The heated metal will slightly melt the plastic, allowing it to deform and form tightly around the metal for a secure fit. This process is done twice, once for each part 5. The press inserts can then be mounted on the 3D-printed parts. It is important to press them straight, otherwise the screws may sit crooked or, in the worst case, not fit at all. Parts 1 and 2 each require four M2 inserts, marked by small holes on the bottom and the side. Part 3 requires two M2 and two M3 inserts: M3 inserts for the threaded rods and M2 inserts for securing the breakout board. Part 8 requires six M2 inserts, three in the back for the MIPI-to-HDMI board, two to hold the side cover, and one in the top for the connection to the IPD mechanism.

With the preparations completed, the assembly can begin. First, the display is slid into the bracket in part 3 and the breakout board is mounted on the back.

Part	Count
3D Printed Parts	10
Threaded Rod M3 55mm	2
Press Insert M3	2
Allen Screw M2 4mm	15
Press Insert M2	16
Display with Breackoutboard	1
MIPI-to-HDMI Board	1

Table 2: Parts for one Eye module

Part	Count
3D Printed Parts	9
Allen Screw M2 4mm	10
Press Insert M2	9
Allen Screw M2 6mm	1

Table 3: Parts for one Eye module

The display and breakout board are then connected through the opening in part 3. Next, part 2 is prepared by placing the lens inside, and part 1 is placed on top of the lens. Part 2 must be positioned below part 1 because it has a small cutout to route the connection cable from the breakout board to the outside. The prepared part 3 is then slid into parts 1 and 2 using the dovetail slides. The connection cable passes through the slit, and part 4 is mounted on the back of parts 1 and 2 to secure the assembly, using four screws. Now, screw both part 5 gears with the threaded rod into the two holes of part 4, which should align with the M3 insert in part 3. Now, screw the gears in until the gears sit flush on part 4. Also, make sure that part 3 is not tilted and is parallel to the lens. Now place part 6 on the middle bump of part 4, and one part 7 on the bump above.

The parts of the surrounding shell are shown in Figure 1 b. This shell encloses the assembled module, blocks out external light, mounts the HDMI-to-MIPI board, and connects the module to the main case. This is the most challenging part of the assembly process.

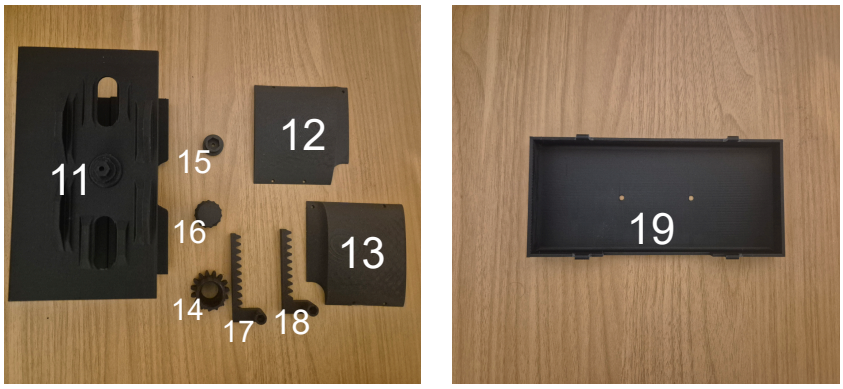
First, part 10 is placed in the smaller compartment in the back of part 8 and secured through the hole with the second part 7 gear. The previously assembled module is then slid into the prepared part 8 with all gears already in place. During sliding, the cable must be routed through the opening on top of part 8. Once the module is almost fully inserted, there is a good chance that the part 7 gear holding part 10 will not align with the gears on part 4. By gently moving part 10 while pressing the assembly together, part 7 will engage correctly with the inner gear system.

Once aligned, the assembly is secured in place by inserting two screws into the top and bottom front of part 8, which are positioned directly above and below the lens. Then Part 9 is placed and screwed onto the side of the module, sealing the shell. For the final step, the HDMI-to-MIPI board is mounted to the back of the assembly, and the cable from the breakout board is connected.

The eye module is now fully assembled and ready for use. To build the module for the opposite side, the same steps are followed, but the process is mirrored.

1.2 Case and Electronics Compartment

In Table 3, the required parts for the case are listed, and the required 3D printed parts are enumerated in Figure 2. As with previous assemblies, the first step is



(a) Case Parts (b) Electronics Compartment

Figure 2: 3D printed Parts for the case and the Electronics Compartment

to prepare the parts for mounting. This means inserting all necessary threaded inserts before doing any other work. In this case, only part 11 requires threaded inserts. Four inserts are needed on each side to mount the side panels, and one insert is placed on top, centered on the bump. With this, the preparation for the case assembly is complete.

First, the two eye modules are inserted into the left and right sides of part 11. To secure them, the left and right panels (parts 12 and 13) are mounted on their respective sides. It is important to check that both modules can move freely and do not bind to the dovetail slides during adjustment.

For the IPD adjustment mechanism, part 17 is placed on top of the device. The higher side is inserted through the left hole and onto the top of the left eye module, where a hexagonal knob serves as the mounting point. The same process is repeated on the right side with part 18. Each connection point is secured with an M2 $\times$ 4 mm screw. After moving both modules so that they meet in the center, part 14 is placed on the center bump at the top of the case, enabling synchronized movement of both modules by turning the gear. Next, part 15 is inserted and fixed using an M2 $\times$ 6 mm screw, which is threaded into the prepared insert—this is the only step that uses a different screw size. Then, part 16 is screwed into part 14 so that, when tightened, it presses against part 15 and creates a friction joint. Because part 15 has a hexagonal connection point, it cannot rotate, which ensures that the entire IPD mechanism is locked in place. Part 19 can be clipped on the backside of the device. It features two holes in which a camera can be mounted. The headset is now fully assembled and can be plugged into a PC with HDMI and USB cables.

2 Compiling Monado for Windows

The first challenge to overcome is compiling Monado for Windows. For this, the Visual Studio 2022 compiler (MSVC) and the CMake build system were

installed and used. For dependency management, VCPKG should be used. It is a cross-platform package manager for C++ dependencies. Fortunately, VCPKG is fully integrated into the project, so CMake handles dependency management. The required dependencies for Windows are illustrated in Table 4.

With this in place, CMake successfully generates the project files, allowing the build process to begin. However, this leads to compilation errors, as the entire project was originally based on a Linux compiler. Unfortunately, the standard libraries differ slightly, which caused the initial compilation to fail due to M_PI being undefined. This issue can be quickly resolved by adding the line: #define M_PI (3.14159265358979323846). After this addition, the compilation process completes successfully, and the system can be started for the first time.

3 Starting the System

By building and executing the "monado_service" target, the service can be started and used. However, at this stage, neither a device instance is created nor is the camera detected correctly. The reason for this is the absence of certain required environment variables (see Table 5).

The first variable is responsible for creating an instance of the custom VR device on startup. This is achieved by disabling automatic probing and implicitly registering the device in the system. The second variable is required for the RealSense driver. To correctly identify the connected cameras, the prober must be assigned a valid source index. Without this, the system crashes during startup due to missing stream configuration parameters.

For the SLAM library, two additional environment variables are necessary. One defines the path to the SLAM library, in this case the Basalt library. The other specifies the configuration file for the D435i. Basalt requires this configuration file to operate correctly, as it contains essential parameters such as camera intrinsics, resolution, IMU biases, standard deviations, and other calibration data. These values can be extracted from the camera itself using the RealSense

Dependecie name	Description
Eigen 3.x	linear algebra library
Pthreads Windows	POSIX threading header for Windows
SDL	access to mouse, keyboard, and other peripherals
Vulkan	Graphic API
HIDAPI	Bluetooth and USB interface
LibUSB	For USB device access
cJSON	JSON parser for C
WIL	For easier Windows development

Table 4: This Table lists the required dependencies for Monado along with a brief description of their general purpose. These dependencies must be installed to allow CMake to generate the necessary build files successfully.

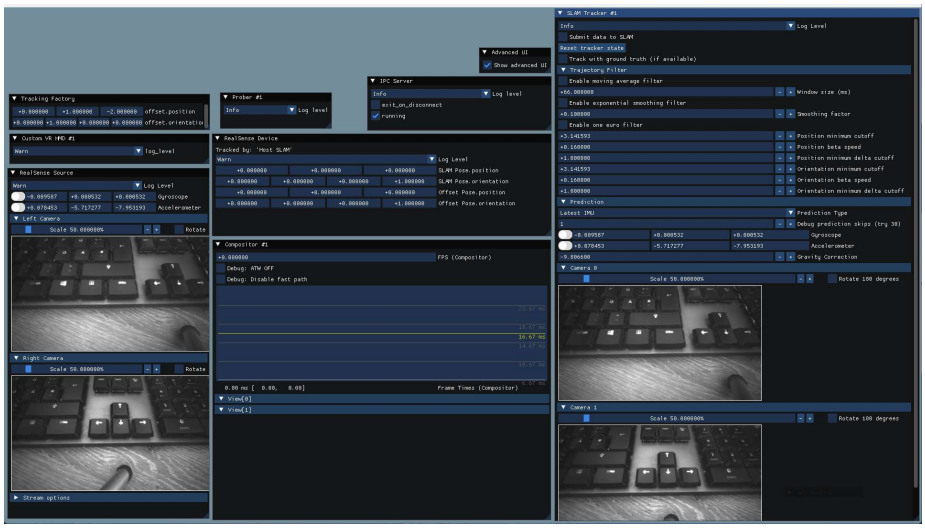


Figure 3: This image shows the monado UI consisting of multiple windows.

Name	Value
CUSTOM_VR_ENABLE	1
RS_SOURCE_INDEX	0
SLAM_CONFIG	./d435i.toml
VIT_SYSTEM_LIBRARY_PATH	./basalt.dll
XRT_DEBUG_GUI	1

Table 5: Environment variables for starting monado properly. The first one is for instantiating the headset. Second for the streaming index of the camera, third and fourth for the slam configuration and the last one to enable the debug gui.

camera calibration tool. For the IMU-specific statistical values, a pre-generated configuration file tailored for the D435i was used. The last environment variable enables the use of the debug GUI, shown in Figure 3. This interface was created using ImuGUI and provides controls and statistics for various features within Monado. Several windows within the UI are handy and have been frequently utilized during development.

4 Configuring Environment

To inform OpenXR runtime that our platform exists and that we intend to use it as our standard value, the registry must be edited. In *Computer\HKEY_LOCAL_MACHINE \SOFTWARE\Khronos\OpenXR\1* the path to the file *openxr_monado-dev.json* must be set. This file is generated during the application build and resides in

the binaries directory. It has all the necessary values for the OpenXR loader to load an OpenXR application with the Monado runtime.

With this setup, it is possible to start VR programs from Unity, which then are displayed by Monado. For Steam gaming, there is a bit more setup involved. First of all, the Steam plugin needs to be compiled.

This is done by compiling the "driver_monado" target. This creates a new file in the *steamvr-monado/bin* including the *monado_driver.dll* which is the plugin for Steam. With this plugin, the driver now needs to be registered on Steam. This is done with a separate batch file that is shipped with the application. `./steam/steamapps/common/SteamVR/bin/vrpathreg.sh adddriver C:/path/to/dll/monado\_driver.dll` let Steam know where the custom VR driver is. The success of the command can be verified by running vrpathreg.sh without any parameters. This enables the use of all hardware supported by Monado on Steam, but does not allow for playing games.

To enable game support, the OpenComposite application must be installed. It is responsible for forwarding the SteamVR calls to OpenXR. wBy starting the program and clicking on the "Switch to OpenVR" button, the redirect is set in place. For the last step, a file in the game needs to be changed to work. Somewhere in the game directory, there is a file called *openvr_api.dll* . The OpenComposite repository provides a replacement file that needs to be swapped out in the game files. With this in place, Monado can be booted up, and then the game can start. This should now open the rendered window. The window can now be placed on top of the two displays to see the rendered view in the headset.