Python Scripts for moving information from Celestica spreadsheets to the ATLAS ITk Production Database

Ruchi Soni

Supervisors: Richard Teuscher and William Trischuk

University of Toronto

September 2022

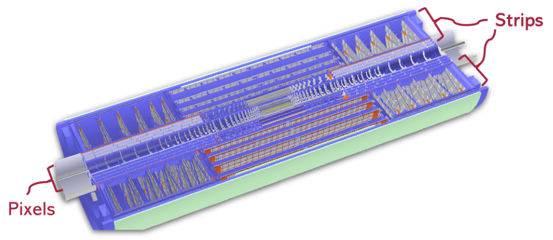
Abstract

The University of Toronto contributes to the development of the Inner Tracker of ATLAS for the High Luminosity Large Hadron Collider. Celestica, a private company, builds the modules and hybrids. Celestica keeps a log of all the assemblies and tests conducted on the components in a Google spreadsheet. This information must also be uploaded onto the ATLAS Inner Tracker Production Database. The database keeps track of the state and history of all the components. To automate the uploading process, six scripts were written to achieve six different tasks: three for uploading tests and three for assembling components. This report will focus on the assembling scripts. One script assembles hybrid flexes to hybrid assemblies, another assembles hybrid assemblies and powerboards to a module, and the third for hybrid assemblies to hybrid test panels. These scripts require minimal interaction with the user and are built to prevent crashes in case of errors. However, the user must be attentive to any changes or rearrangements made in the spreadsheet. The scripts may need to be modified accordingly to avoid errors. Otherwise, the scripts prove to be an effective method to transfer information.

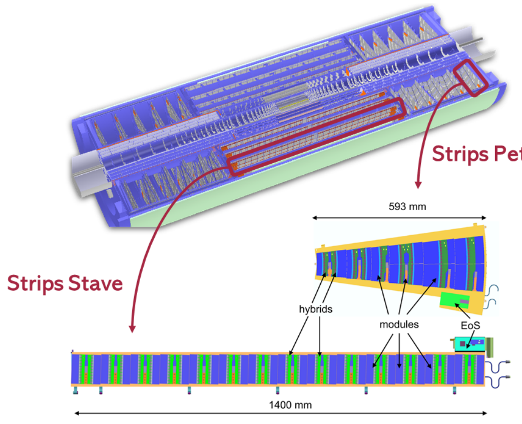
1 Introduction

The Large Hadron Collider (LHC) currently collides protons at an energy of 13.6 TeV with an instantaneous luminosity of 2×10^{34} cm^{-2}s^{-1} (Calderini, 2022). To increase the potential for discovering new physics, physicists require many more collisions to analyze. The High Luminosity LHC (HL-LHC) project is thus developed. The LHC will undergo a major upgrade which will result in an integrated luminosity of 4000 fb^{-1} over ten years. The upgrade will result in a pile-up of around 200 collisions per bunch crossing compared to the current 40 collisions per bunch crossing. The increase in luminosity naturally requires a detector that can survive much more radiation. The detectors must be able to keep up with the increased radiation to take full advantage of the high luminosity. They should therefore have faster readout channels and higher granularity.

The ATLAS detector at CERN will receive upgrades to the readout systems of the calorimeters and the muon spectrometer, as well as the trigger system to combat the degradation caused by the radiation. The current Inner Detector of ATLAS will be replaced with the new Inner Tracker (ITk) containing all-silicon pixel and strip subsystems, as illustrated in Figure 1a. The pixel system is at a smaller radius and is the first line of detection for charged particles emerging from a collision. As a result, they have a much higher resolution for particle detection. The strips system surrounds the pixels and covers a larger area. It consists of five layers of barrels which run longitudinally in the middle, and six endcap disks on each side of the barrels (Calderini, 2022). The barrels are composed of staves and the endcaps of petals, as depicted in Figure 1b.



(a)



(b)

Figure 1: The cross section of the ITk. (a) The pixel and strips subsystems and (b) a zoom into the barrel stave and endcap petal with their location in the ITk. Adapted from ATLAS Inner Tracker Strip Detector Technical Design Report by The ATLAS Collaboration.

The basic building block of the ITk is the module, shown in Figure 2. Modules are built differently for the pixels and strips and for their corresponding staves and petals. However, they have the same basic components: silicon sensor, hybrid, powerboard and readout chips (The ATLAS Collaboration, 2017). A hybrid consists of the green piece with the readout chips glued on the top.

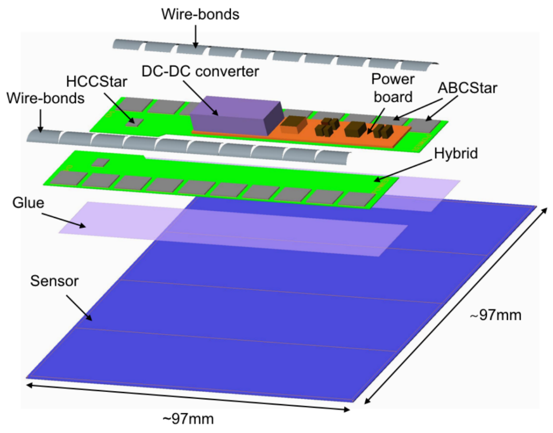


Figure 2: A short strip barrel module showing the different components that a module is comprised of. Adapted from ATLAS Inner Tracker Strip Detector Technical Design Report by The ATLAS Collaboration.

The University of Toronto (U of T) works on the endcap strip modules. The endcap petal consists of 6 different module types named R# where # ranges from 0 to 5. U of T focuses on R0 and R3 module types and all hybrid types. Depending on the type of module, the number of hybrids and powerboards varies for endcaps, as indicated in Table 1. A module is split into two half-modules for R3, R4 and R5 types. The detailed layout of the modules, half-modules, and hybrids in the petal is shown in Figure 3.

Table 1: The number of hybrids and powerboards for each different module/half-module type in a petal.

Type	½ Mod.	# Hyb.	# PWB
R0	-	2	1
R1	-	2	1
R2	-	1	1
R3	R3M0	2	1
	R3M1	2	0
R4	R4M0	1	1
	R4M1	1	0
R5	R5M0	1	1
	R5M1	1	0

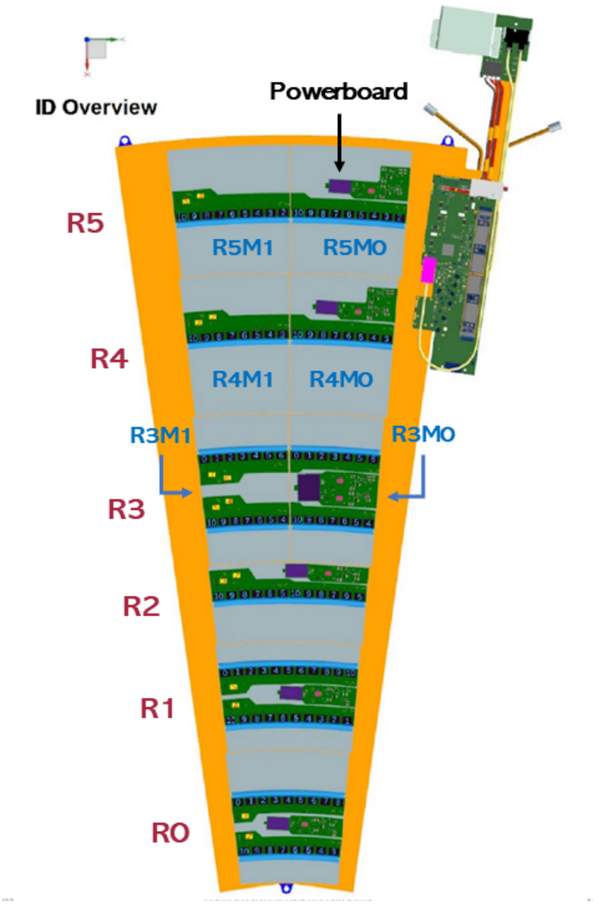


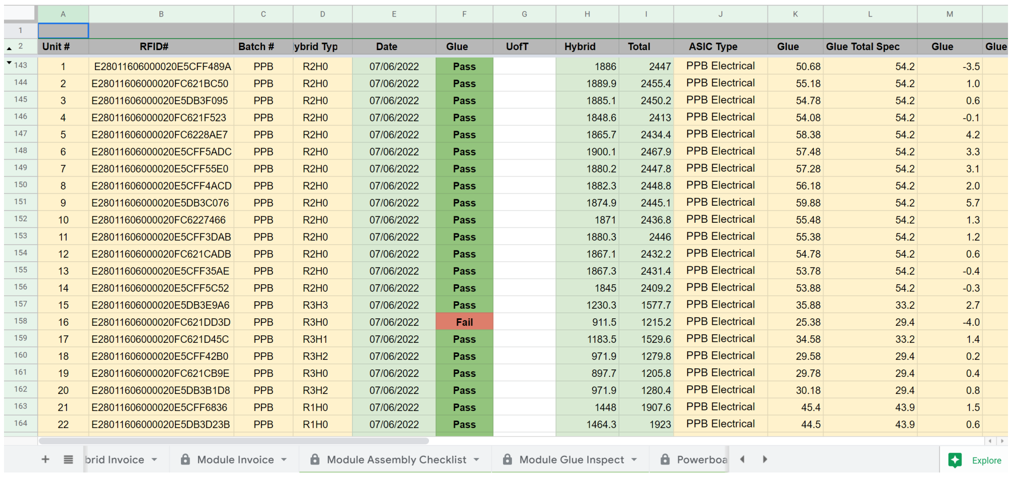
Figure 3: The strips endcap petal labelled with the module types (including full and half-modules). Adapted from Endcap Hybrid Wire-Bond Documentation STAR Chipset V2.1 by William Trichuk et al.

The U of T contracts Celestica, an electronics manufacturing company, to build the hybrids and modules. Celestica must perform several tests on these components to ensure their functionality. All the information about assembling various parts and testing them is recorded in a Google Spreadsheet by Celestica and is shared with a few on the U of T ATLAS group.

1.1 ITk Production Database

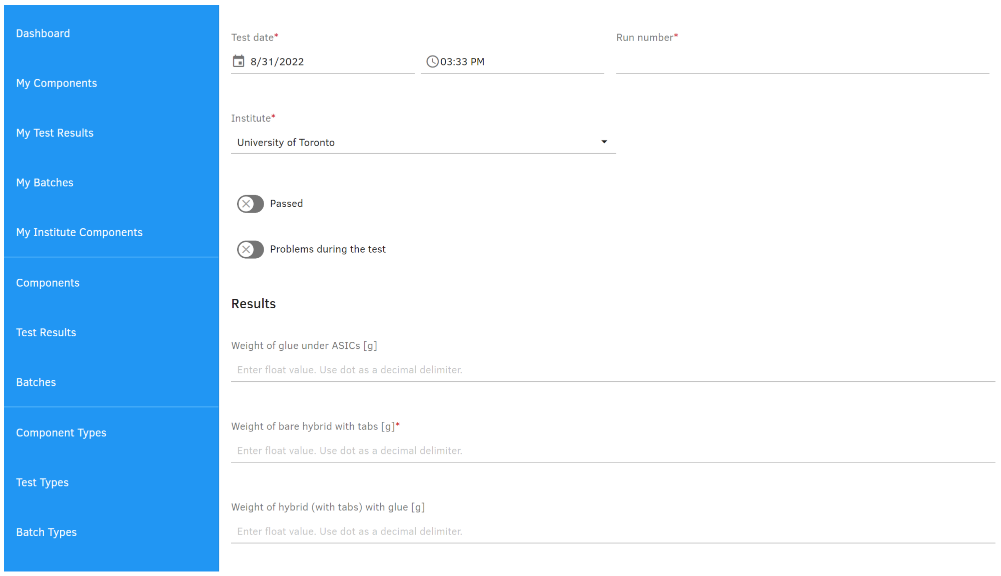
The ATLAS collaboration has designed a production database for the ITk upgrades that contains all the relevant information regarding a component. When various components are assembled or disassembled physically, they must also be "assembled" or "disassembled" in the database. Each component has a unique identification number, as well as a code. The filled-in Celestica spreadsheets hold information that must be uploaded to the ITk database. The database contains the following general information about each component: a non-unique serial number, an alternative identifier, the type of the component (e.g. module, hybrid panel), the type (e.g. R0, R1), the stage it is currently at, its current location, its home institute where it was first registered, its properties (different by component type), a list of parents and children, and a list of tests conducted on it. The history of the stages, the history of the shipments, and the history of the component's previous parents and children are also available.

The ITk database names components differently. A hybrid is split into a hybrid flex and a hybrid assembly. The hybrid flex has an RFID number identifying it and is physically listed on the green unit itself. It is a hybrid without any chips on it. The hybrid assembly is the parent of the hybrid flex, as well as of the other chips (ABCStar and HCCStar)[1]. The hybrid assembly can then be assembled to either a hybrid test panel or a module. A hybrid panel is used to transport multiple hybrids from one location to another before assembling them to a module. In addition to a hybrid assembly, a module also has a sensor and a powerboard for children. If the module type is R3, R4 or R5, it is split into two half-modules. Then, each half-module may be composed of a hybrid assembly, sensor and powerboard instead, and the two half-modules are assembled to a full module. The module has a module frame as a parent.



	Unit #	RFID#	Batch #	ybrid Typ	Date	Glue	UofT	Hybrid	Total	ASIC Type	Glue	Glue Total Spec	Glue	Glue
143	1	E28011606000020E5CFF489A	PPB	R2H0	07/06/2022	Pass		1886	2447	PPB Electrical	50.68	54.2	-3.5	
144	2	E28011606000020FC621BC50	PPB	R2H0	07/06/2022	Pass		1889.9	2455.4	PPB Electrical	55.18	54.2	1.0	
145	3	E28011606000020E5DB3F095	PPB	R2H0	07/06/2022	Pass		1885.1	2450.2	PPB Electrical	54.78	54.2	0.6	
146	4	E28011606000020FC621F523	PPB	R2H0	07/06/2022	Pass		1848.6	2413	PPB Electrical	54.08	54.2	-0.1	
147	5	E28011606000020FC6228AE7	PPB	R2H0	07/06/2022	Pass		1865.7	2434.4	PPB Electrical	58.38	54.2	4.2	
148	6	E28011606000020E5CFF5ADC	PPB	R2H0	07/06/2022	Pass		1900.1	2467.9	PPB Electrical	57.48	54.2	3.3	
149	7	E28011606000020E5CFF55E0	PPB	R2H0	07/06/2022	Pass		1880.2	2447.8	PPB Electrical	57.28	54.2	3.1	
150	8	E28011606000020E5CFF4ACD	PPB	R2H0	07/06/2022	Pass		1882.3	2448.8	PPB Electrical	56.18	54.2	2.0	
151	9	E28011606000020E5DB3C076	PPB	R2H0	07/06/2022	Pass		1874.9	2445.1	PPB Electrical	59.88	54.2	5.7	
152	10	E28011606000020FC6227466	PPB	R2H0	07/06/2022	Pass		1871	2436.8	PPB Electrical	55.48	54.2	1.3	
153	11	E28011606000020E5CFF3DAB	PPB	R2H0	07/06/2022	Pass		1880.3	2446	PPB Electrical	55.38	54.2	1.2	
154	12	E28011606000020FC621CADB	PPB	R2H0	07/06/2022	Pass		1867.1	2432.2	PPB Electrical	54.78	54.2	0.6	
155	13	E28011606000020E5CFF35AE	PPB	R2H0	07/06/2022	Pass		1867.3	2431.4	PPB Electrical	53.78	54.2	-0.4	
156	14	E28011606000020E5CFF5C52	PPB	R2H0	07/06/2022	Pass		1845	2409.2	PPB Electrical	53.88	54.2	-0.3	
157	15	E28011606000020E5DB3E9A6	PPB	R3H3	07/06/2022	Pass		1230.3	1577.7	PPB Electrical	35.88	33.2	2.7	
158	16	E28011606000020FC621DD3D	PPB	R3H0	07/06/2022	Fail		911.5	1215.2	PPB Electrical	25.38	29.4	-4.0	
159	17	E28011606000020FC621D45C	PPB	R3H1	07/06/2022	Pass		1183.5	1529.6	PPB Electrical	34.58	33.2	1.4	
160	18	E28011606000020E5CFF42B0	PPB	R3H2	07/06/2022	Pass		971.9	1279.8	PPB Electrical	29.58	29.4	0.2	
161	19	E28011606000020FC621CB9E	PPB	R3H0	07/06/2022	Pass		897.7	1205.8	PPB Electrical	29.78	29.4	0.4	
162	20	E28011606000020E5DB3B1D8	PPB	R3H2	07/06/2022	Pass		971.9	1280.4	PPB Electrical	30.18	29.4	0.8	
163	21	E28011606000020E5CFF6836	PPB	R1H0	07/06/2022	Pass		1448	1907.6	PPB Electrical	45.4	43.9	1.5	
164	22	E28011606000020E5DB3D23B	PPB	R1H0	07/06/2022	Pass		1464.3	1923	PPB Electrical	44.5	43.9	0.6	

(a)



Test date*
8/31/2022
03:33 PM

Run number*

Institute*
University of Toronto

Passed

Problems during the test

Results

Weight of glue under ASICs [g]
Enter float value. Use dot as a decimal delimiter.

Weight of bare hybrid with tabs [g]*
Enter float value. Use dot as a decimal delimiter.

Weight of hybrid (with tabs) with glue [g]
Enter float value. Use dot as a decimal delimiter.

(b)

Figure 4: (a) An example of a Celestica spreadsheet. This one shows the Hybrid Weight sheet. (b) Interface of the ITk database, showing the glue weight test upload page. Information from (a) must be recorded in (b).

[1]ABC stands for Analog to Binary Converter and HCC stands for Hybrid Control Chip.

2 Motivation

Previously, the ATLAS group at U of T manually filled in the information from the Celestica spreadsheet (Figure 4a) to the ITk database using the database interface (Figure 4b).

With the growing number of hybrids and modules assembled and tested by Celestica, manually entering the information about all the components becomes inefficient and tedious. Therefore, it becomes worthwhile to automate this process using code. The idea was to have a script where the user only needs to input the rows they would like to work on. The task itself is performed by the script on the selected rows. The transferring of information is then much faster and productive. The objective of this report is to describe the properties, usage, advantages, and drawbacks of the scripts that are written.

3 Product

There are six scripts designed using python for six different tasks:

- `assemble_flex_to_assembly.py`
- `assemble_hybrid_to_module.py`
- `assemble_hybrid_to_panel.py`
- `hybrid_glue_weight.py`
- `module_glue_weight.py`
- `hybrid_wirebonds.py`

The first three scripts are for assembling one component to another, and the last three are for uploading tests for a given component. The three tests are hybrid and module glue weight tests and hybrid wirebond tests. This report will focus on the assembling scripts. The six scripts and other relevant files are shared in a GitLab repository where it is accessible to all CERN members.

The `assemble_flex_to_assembly.py` script takes the RFID of the hybrid flex listed in the spreadsheet and searches for that flex in the database. This script assumes that the flex is already registered by the institute from where it was shipped. If the hybrid flex is already assembled to a hybrid assembly, the script moves to the next row. If the flex is assembled to a hybrid flex array, it is disassembled from it. In this case and in the case where the flex has no parents, a new hybrid assembly is registered. The hybrid flex is then assembled to the new hybrid assembly.

The `assemble_hybrid_to_module.py` script assembles hybrid assemblies and powerboards to the module. This script does not register any components. All the components (modules, hybrid assemblies, module test frames, panels, flexes and sensors) are assumed to be registered. However, the module's identifying code is not provided in the spreadsheet. Therefore, the module code is obtained by searching for the sensor's parent as they are assumed to be assembled earlier. The sensor's ID is first acquired by its given lot number. If the sensor is found to have a parent with a type different than a module or has no parent at all, the row will be skipped entirely. Each row in the spreadsheet corresponds to either a module or half-module and can have up to two hybrids. The script loops through each hybrid and the powerboard (if there is one). First, the script looks for their parent. If the parent is the same module as the sensor's parent, it's already assembled, and nothing needs to be done. If the sensor and assembly are assembled to two different modules, an error is printed, and the row is skipped. If the parent of the hybrid assembly or powerboard is a hybrid test panel, they are disassembled and assembled to the module (sensor's parent). The serial number of the powerboard is first constructed using the powerboard number given in the spreadsheet and a powerboard version code to search for it in the database. If the powerboard version changes, the constant `POWERBOARD_VERSION` defined at the beginning of the script's code must be modified. Before the assembly happens, the component's assembly properties are required. These properties include the jig used for hybrid/powerboard alignment, the pickup tool used, the date of the glue sample, and the jig for module assembly. The required information is in the spreadsheet and is simply retrieved by the script. The RFIDs given for the jigs and pickup tools are converted into serial numbers for easier search through the database. If the hybrid assembly is successfully assembled to the module, its stage is updated to `ON_MODULE`. If the powerboard is successfully assembled to the module, its stage is updated to `LOADED`. Lastly, if all the hybrids (and the powerboard) are assembled to the module, the module is upstaged to `GLUED`.

The `assemble_hybrid_to_panel.py` script assembles hybrid assemblies to hybrid test panels. The panels are designed to contain a maximum of four hybrids and two powerboards. The sheet for this task is arranged such that each row is a different hybrid flex with four consecutive hybrids having the same hybrid panel. Therefore, for all panel types except for R3, each hybrid also has a corresponding position on the panel to differentiate between them. The R3 already has four different hybrids, so the position is not required to distinguish them. The script assigns positions (from 0 to 3) to hybrids based on the order they appear in the spreadsheet. For example, an R0 Test Panel's hybrids appear as R0H0, R0H1, R0H0, and R0H1 in the spreadsheet; their positions would then be 0, 1, 2, and 3, respectively. The assembling also requires two other properties, the jig and pickup tool used for placing the hybrid on the panel. This information is found on a different spreadsheet called "Tools List." In the end, if all hybrids are successfully assembled, the panel's stage is updated to `HYBRIDS_ASSEMBLED`.

3.1 Properties of the Scripts

All the scripts are designed to not redo operations on a row if previously executed. The scripts work for all hybrid and module types.

3.1.1 `helper_functions.py`

The file `helper_functions.py` is composed of multiple general functions that are used in the scripts. It is a python module, and its function is to be imported into the six scripts. The file also contains some additional functions that have not been used in any script; however, they may be applicable during debugging.

3.1.2 Service Account

A python library called `gspread` was used to read the Google spreadsheet. The Celestica spreadsheet is not accessible to everyone. Viewing access was specifically granted by Celestica to an email address for the scripts to read data from the spreadsheet. The relevant information required for `gspread` to read the spreadsheets, including the email, is stored in a `json` file named `service-account.json`. The purpose of `service-account.json` is to simply be imported into the scripts for `gspread` to use.

3.1.3 Progress File

After running a particular script on a set of rows several times, it may become difficult to keep track of the rows that were run previously. Failure to track the rows can be a problem, especially if multiple users run the script. To make the script faster and to keep a log of the rows that have successfully been processed, each script has its text file called a progress file which records such rows. The naming convention used for progress files is `<SCRIPT_NAME>_progress.txt`. When a script successfully runs its operation on a row without errors, the row number is written to the corresponding progress file. The next time the script is run, there will be an option to skip all the rows that appear in the progress file. Note, if an error occurs such that the task is not fully completed for a row and the row is skipped, that row will not be added to the progress file.

3.2 Usage of the Scripts

Each script interacts with the user in the same manner. First, the script will ask the user to input the row where they want to start running the script. The row here corresponds to the row of the Google Spreadsheet itself (the leftmost column in Figure 4a), not the Unit # given by Celestica. Then, the user will be asked for the row where they would like to end processing. To go to the very last row in the spreadsheet, the option to write “end” as a response is also available. Next, the user has the choice to use the progress file for that script. If the input is yes, the rows in the progress file will be skipped. If the user would like to skip any additional rows between the start and end rows, the option to do so appears afterwards. The skipped rows should be written with spaces separating them. Lastly, the user will be asked for their two access codes for the ITk database to access and make changes in the database.

3.3 Limitations of the Scripts

The scripts written rely highly on how Celestica arranges the columns of their spreadsheets. Whether a column number corresponds to the hybrid flex RFID or whether it refers to the type of hybrid is hard-coded. If Celestica decides to add another column or delete a column, the position of other columns changes. A similar problem occurs if they switch one column with another. For example, column 1, which is for hybrid RFID, could now refer to the hybrid type. When the script tries to perform the task with information from this column, it will most likely encounter an error. Therefore, if a change is made with the columns in a sheet, then the column indices must be modified in the script’s code. The column indices are defined as constants at the beginning of the script in upper case letters. Note: the column indices start counting from 0 (that is, column A corresponds to 0). One should also be watchful of typing errors in the spreadsheet. The row positions should ideally not be modified after running a script, as this could lead to inconsistencies in the progress file.

The name of the various spreadsheets should also not be changed. The spreadsheet used for each script is defined at the start of the code as `G_SHEET_NAME`. This constant must be modified if a sheet name change is made. If the spreadsheets use any date convention other than DD-MM-YYYY or DD/MM/YYYY, then the result in the database could be wrong, or the script could encounter an error.

The scripts are constructed to not quit in case of errors. However, the script can crash if there are typos when entering access codes or an unexpected error from the database (a common error is code 500 when the database is down). In this case, the user should first try to rerun the script or run it after a few minutes.

4 Conclusion

Institutes from all over the world are contributing to the ATLAS upgrades for the High-Luminosity LHC. The information filled in by Celestica must be transferred to the ATLAS ITk Production database. A set of python scripts allows U of T to do the transfer automatically. The six scripts created each execute a different task: uploading hybrid/module glue weight test and hybrid wirebond test, assembling hybrid flex to hybrid assembly, hybrid assemblies and powerboards to modules, and hybrid assemblies to panels. These scripts require minimum input from the user. In case of errors, they skip the row to prevent a crash. However, these scripts are highly dependent on the state of the Celestica spreadsheets. In case of any modifications to the spreadsheet, the code must be edited accordingly. Overall, the scripts are user-friendly and efficient, making it easier and more productive to keep the ITk database updated. In the future, more scripts can be written as needed. This is not a hard task since much of the code is the same across the scripts.

5 Acknowledgements

A big thank you to Leandro Rizk, the co-author of the scripts, for his invaluable input and support throughout this fellowship. I would also like to acknowledge my supervisors, William Trichuk and Richard Teuscher, for the wealth of information they have provided regarding this project and otherwise. The guidance of the U of T ATLAS group is highly appreciated; a special thanks to Laurelle Veloce for her continued assistance with the formation of the scripts. Lastly, I thank the Institute of Particle Physics and CERN for organizing the Summer Student Programme and providing us with the opportunity to expand our learning.

References

Calderini, G. (2022). The atlas itk detector for high luminosity lhc upgrade. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, *1040*, 167048. https://doi.org/10.1016/j.nima.2022.167048

The ATLAS Collaboration. (2017). Technical design report for the atlas inner tracker strip detector.