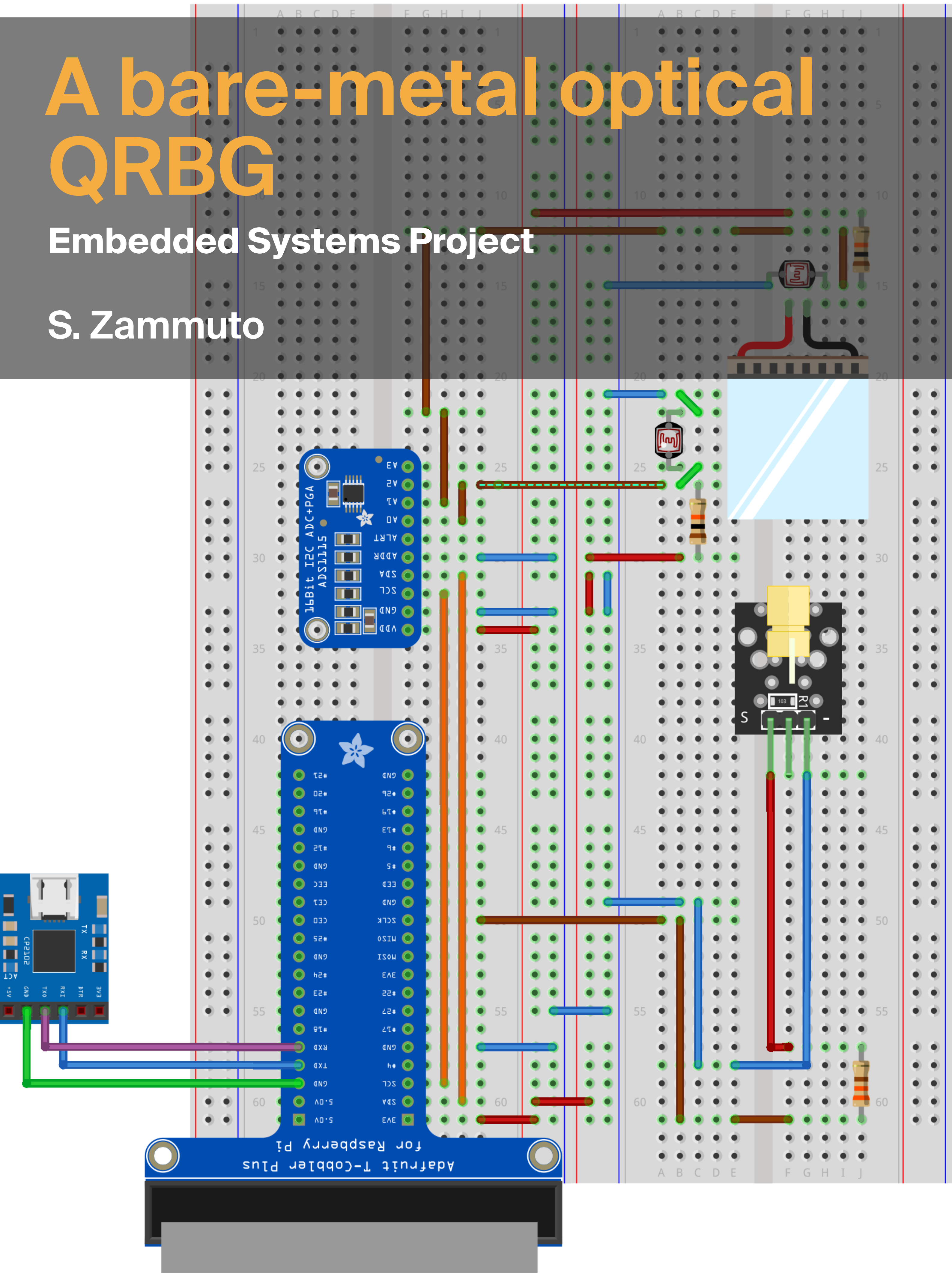
A bare-metal optical QRBG

Embedded Systems Project

S. Zammuto



A bare-metal optical QRBG

Embedded Systems Project

by

S. Zammuto

—

Supervisor:	Prof. D. Peri
Faculty:	Department of Computer Engineering
University:	University of Palermo

Cover: Breadboad schematics of the QRBG

Summary

This document contains the description of an Embedded Systems project realized for the Master's Degree Course in Computer Engineering at the University of Palermo.

The project focuses on the design and implementation of a Quantum Random Bit Generator (QRBG) controlled by a single-board controller that is interfaced in bare-metal mode through Forth, a lightweight, stack-based, intermediate-level language that provides an efficient management for both computational and communication tasks.

Random bit generation is the process of producing bits that exhibit no discernible pattern or predictability, thereby appearing random. These bits are crucial for a wide array of applications across various domains, including cryptography, simulation, gaming, and secure communication protocols. Traditional methods of random bit generation, such as pseudo-random number generators (PRNGs), rely on deterministic algorithms to produce sequences of numbers that approximate randomness. However, these sequences are ultimately predictable, which can pose security risks in certain applications.

With the recent advent of Quantum Technologies, which, among all of the other things, promise to achieve such significant computational speed-ups so as to render classical, brute-force resistant cryptography obsolete, concerns arise for the security of communications. On the flipside, those very same technologies that pose a threat to the integrity of the users' data, can be exploited to design and build mechanisms that are resistant to those point of attack.

Out of the different hardware technologies that implement the quantum paradigm, the optical/photonic one is the alternative that, due to its robustness and ease of implementation, appears to be most suitable for dealing with the exchange of information between two or more parties. It is upon those consideration that this project achieves Random Bit Generation through the realization of a laser-based physical qubit, the information unit of quantum computation, along with the (classical) embedded system that is required to control it.

Tests on the randomness of this toy system showcase the maximal entropy of the integrated half-interferometer system, proving the validity of the optical approach from both a quality and performance of one such technology.

Contents

Nomenclature

Abbreviations

Abbreviation	Definition
ADC	Analog-to-Digital Converter
BB	Breadboard
BSC	Broadcom Serial Control
Δ/Σ	Delta-Sigma Modulation
GND	Ground
GPIO	General-Purpose Input/Output
I^2C	Inter-Integrated Circuit
IC	Integrated Circuit
ISA	Instruction Set Architecture
LD	Laser Diode
LDR	Light Dependent Resistor
ML	Machine Language
MUX	Multiplexer
PGA	Programmable Gain Amplifier
PRNG	Pseudo-Random Number Generator
QRBG	Quantum Random Bit Generator
SCL	Serial Clock
SDA	Serial Data
SPS	Samples per Second
SBC	Single-Board Computer
SoC	System-on-a-Chip
TRNG	True Random Number Generator
TTL	Transistor-Transistor Logic
TX/RX	Transmit/Receive Signals
UART	Universal Asynchronous Receiver/Transmitter
VDD	Voltage at Drain

Symbols

Symbol	Definition
$\vert\psi\rangle$	State Vector
$\vert 0\rangle$ $\vert 1\rangle$	Basis States
H	Hadamard Operator

1 Introduction

The objective of this work is to build and program the embedded controller of a Random Bit Generator based on the Linear Quantum Optics paradigm. It is worth noting that, while the core optical mechanism is what ultimately determines the quality of achieved randomness, one such apparatus cannot exist without a classical controller that interfaces it for managing inputs and readouts, which is indeed the entire purpose of Embedded Systems.

For this reason, a precise and efficient configuration is mandatory if practical usefulness is desired to be at least mimicked. Here, the main controller is exploited in a bare-metal fashion, directly interfacing the peripherals of its inner chip with the idea of both maximizing performance and minimizing resource utilization.

1.1. Random Bit Generation

Random bit generation, a fundamental aspect of modern computing, involves the creation of sequences of bits that exhibit no discernible pattern or predictability. These bits serve as the backbone for a plethora of applications spanning various domains, including cryptography, simulations, and secure communications. While traditional methods such as pseudo-random number generators (PRNGs) rely on deterministic algorithms to approximate randomness, they may still retain discernible patterns that can potentially compromise security.

Random bit generation is indispensable for ensuring the security, privacy, and reliability of computational systems across industries. In cryptographic applications, random bits are pivotal for generating cryptographic keys, enabling secure authentication mechanisms, and creating unique session tokens for safeguarding communication channels. Similarly, in simulations and gaming, the generation of random bits facilitates the creation of unpredictable outcomes, thereby enhancing user engagement and experience.

1.2. Going Quantum

Quantum random bit generation represents a paradigm shift in the realm of randomness generation, offering a plethora of advantages over conventional methods:

- **True Randomness**: quantum random bit generators harness the intrinsic randomness of quantum phenomena, such as the probabilistic nature of quantum measurements or the randomness inherent in quantum fluctuations. Unlike their deterministic counterparts, quantum random bit generators produce bits that are truly random and inherently unpredictable, thereby offering a higher degree of security and robustness.
- **Uniformity and Unbiased Distribution**: such bit generators yield bits with a uniform distribution, ensuring that each bit has an equal probability of being either 0 or 1. This characteristic

is paramount in cryptographic applications, where biased distributions can compromise the integrity of cryptographic keys and algorithms.

- **Elevated Security Posture**: the unpredictability of quantum random bits enhances the security posture of computational systems by thwarting attacks predicated on statistical analysis or pattern recognition. Quantum random bit generators provide cryptographic strength that is impervious to deterministic algorithms, thereby fortifying the resilience of cryptographic protocols and communication channels.
- **Future-Proofing in an Evolving Landscape**: quantum random bit generation offers a future-proof solution to the burgeoning demands of security in emerging technologies such as quantum computing and the Internet of Things (IoT). As computing power escalates and adversaries devise novel attack vectors, quantum random bit generators serve as a cornerstone for ensuring security and privacy in an increasingly interconnected and digitally-driven world.

1.3. Optical Technologies

The KLM (Knill-Laflamme-Milburn) protocol stands as a pioneering framework that elucidates how optical quantum computing can be practically realized. Leveraging the inherent properties of photons, the KLM protocol outlines a methodology for implementing quantum gates and operations crucial for quantum computation.

At its core, optical quantum computing harnesses the quantum nature of photons, exploiting their ability to exist in multiple states simultaneously, a phenomenon known as superposition. Moreover, photons possess another characteristic vital for quantum computing: entanglement. Through entanglement, the state of one photon becomes intimately linked with the state of another, regardless of the spatial separation between them. This phenomenon enables the creation of qubits, the fundamental units of quantum information.

The KLM protocol provides a blueprint for realizing quantum gates, the building blocks of quantum circuits, using these properties of photons. One of the key components of the KLM protocol is the use of linear optical elements, such as beam splitters and phase shifters, to manipulate the quantum states of photons. These elements enable operations like quantum interference and probabilistic quantum measurements, which are essential for implementing quantum gates.

Furthermore, the KLM protocol elucidates the concept of fault-tolerant quantum computing in the optical domain. It outlines strategies for error correction and mitigation, essential for overcoming the inherent fragility of quantum states in practical quantum computing systems.

Overall, the KLM protocol serves as a cornerstone for optical quantum computing, offering insights into the implementation of quantum gates, error correction, and fault tolerance. Through its meticulous framework, the KLM protocol paves the way for the realization of scalable and reliable quantum computing systems based on optical technologies, promising transformative advances in computational power and capability.

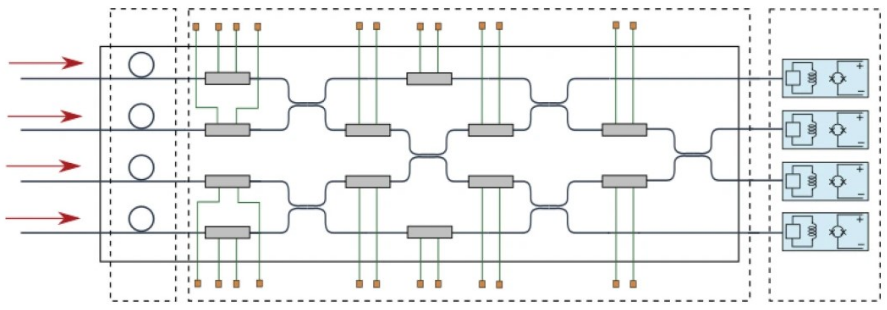


Figure 1.1: Typical hardware layout of a Photonic Quantum Processor.

1.4. The Integrated QRBG System

The following sections describe the actual implementation of the QRBG system. The main controller used is a Raspberry Pi SBC connected to a breadboard where the rest of the components are laid out.

On the breadboard, an half-interferometer configuration is built, where a Laser Diode is activated to shine a ray of photons towards a Beam Splitter prism, which routes the beam towards two Light-Dependent Resistors. When exiting the LD, the quantum nature of the photons is such that they are all in superposition of states, the states being the two main polarization directions (horizontal-vertical) of the light wave.

Then, the voltages on each of those resistors is read, causing the wavefunction of the polarization states to collapse into one of the two directions, and a bit is generated according to which LDR receives more photons, corresponding to the preferential polarization direction chosen by the photons at measurement. This setup is displayed in Figure 1.2.

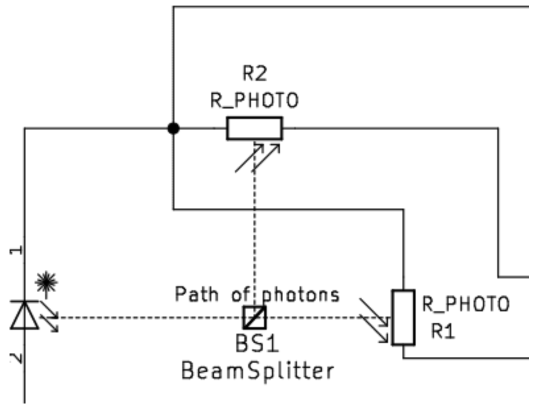


Figure 1.2: Half-interferometer configuration for bit generation.

To put it another way, this configuration realizes a fundamental quantum gate what known as an Hadamard Gate (Fig. 1.3), denoted with H, which is usually used to create superposition states. If $|0\rangle$ and $|1\rangle$ are the basis states, each corresponding to a polarization direction, the Hadamard prepares a new, combined state of the form

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

with $\alpha = \beta = \frac{1}{\sqrt{2}}$ being the complex amplitudes whose squares represent the probability of measuring the combined state into one of the basis ones ($\alpha^2 + \beta^2 = 1$ always).

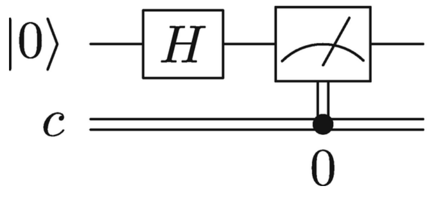


Figure 1.3: Representation of an Hadamard gate with measure according to the quantum-circuit model. At the bottom, a classical bit is included to store the readout of measurement.

Since the Hadamard creates and uniform superposition, that is, it results in a state where $\alpha = \beta$, there will be a 50/50 chance of measuring either of the two states, effectively creating a source of pure, uniform randomness, which is what we look for in a RBG.

2
Tools

Here, all of the components used to build this project are described, while, in the following section, their layout and configuration is explicited. Regular resistors, as well as wire extensions are excluded from this list, as their description is trivial.

2.1. Main Controller

The main controller of this architecture is a Raspberry Pi 4B Single-Board Computer, mounting the RPi JonesFORTH O/S, a bare-metal operating system based on JonesForth-ARM. While the RPi indeed constitutes a general-purpose device, capable of running a full-fledged operating system, the bare-metal mode allows to directly manipulate hardware registers and functionalities, enabling an efficient and precise manipulation of its resources, aligning with the philosophy of special-purpose embedded systems.

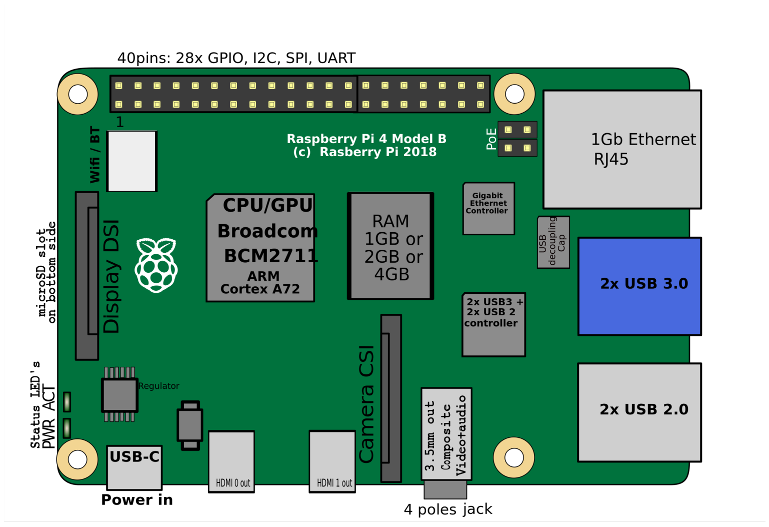


Figure 2.1: Layout of the Raspberry Pi 4B SBC.

This system mounts the quad-core Cortex-A72, 1 GHz, Broadcom BCM2711 as a main chip, built upon the 32-bit ARM Instruction Set Architecture. It is the BCM2711 and its register and components that are directly programmed in order to interface and control the components that implement the BRG logic. Out of the multiple functionalities of this device, those that are employed to achieve our goals are:

- **General-Purpose Input/Output (GPIO)**: GPIOs refers to the pins that can be programmed to serve multiple purposes, including both input and output functionalities (Figure 2.2). These pins allow the Raspberry Pi to interact with the physical world by connecting to various electronic components such as sensors, LEDs, motors, and more. In this instance, GPIOs control the laser, handle the readout values, and also provide a $3.3\ V$ power supply to all of the components on the breadboard.

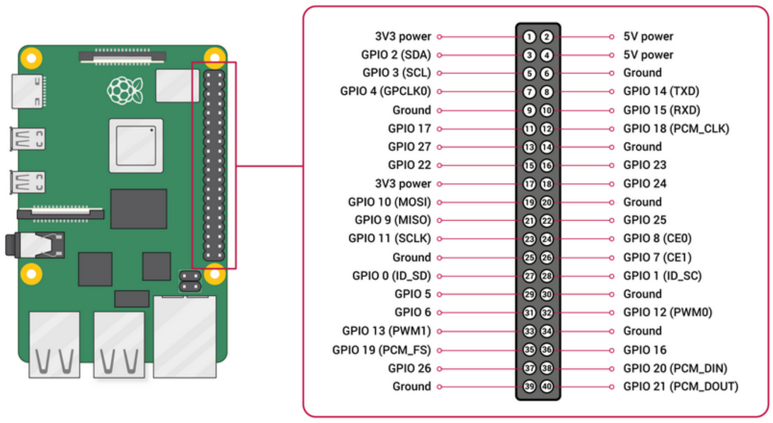


Figure 2.2: 40-Pin Header layout of the Raspberry Pi.

- **Broadcom Serial Control** (**BSC**): internal controller for Inter-Integrated Circuit (I^{2}C) communication with other master/slave devices.
- **System Timer**: hardware timer integrated into the SoC providing accurate timekeeping and event triggering capabilities essential for various system operations.

2.2. Hardware Controller interface

As already mentioned, the main controller is programmed in Forth through pijFORTHos, allowing for efficient data manipulation and processing at the hardware level. To ease the development of the project, code was developed on a standard laptop computer and then uploaded onto the controller through a CP2102 USB-to-TTL module with the UART protocol, interfaced through the ZOC8 terminal tool.

The CP2102 USB to TTL module (Figure 2.3) is a popular and versatile USB-to-serial converter module based on the CP2102 integrated circuit (IC) manufactured by Silicon Labs. It serves as an interface between USB-equipped devices, such as computers or microcontrollers, and devices using TTL-level serial communication, such as microcontrollers, sensors, and other electronic modules.

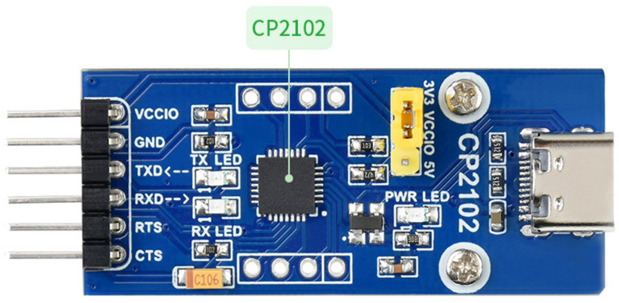


Figure 2.3: CP2102 USB-to-TTL module for UART communication.

Key features and specifications of the CP2102 USB to TTL module include:

- **CP2102 IC**: the module integrates the CP2102 USB-to-serial bridge IC, which provides a USB 2.0 full-speed interface for communication with the host computer and TTL-level serial communication for interfacing with external devices.

- **USB Connectivity**: the module features a USB Type-A connector for connecting to the host computer or USB-enabled device. It is typically plug-and-play compatible with most operating systems, requiring minimal or no driver installation for basic functionality.
- **Serial Communication**: the CP2102 IC supports asynchronous serial communication (UART) with configurable data rates ranging from 300 bps to 3 Mbps. It provides a standard UART interface with transmit (TX), receive (RX), and optional handshake (RTS/CTS) signals.
- **TTL-Level Outputs**: the module typically provides TTL-level serial outputs compatible with 3.3V or 5V logic levels, making it compatible with a wide range of microcontrollers and electronic devices.
- **LED Indicators**: the module typically provides TTL-level serial outputs compatible with 3.3V or 5V logic levels, making it compatible with a wide range of microcontrollers and electronic devices.
- **Software Support**: Silicon Labs provides software drivers and utilities for configuring and interfacing with the CP2102 IC on different operating systems, including Windows, macOS, Linux, and various embedded platforms.

2.3. Laser Diode

A laser diode is a semiconductor device that emits coherent light through the process of stimulated emission. It operates similarly to a light-emitting diode (LED) but with a key difference: while an LED emits incoherent light over a broad spectrum of wavelengths, a laser diode emits coherent light with a narrow spectrum, typically confined to a single wavelength or a very narrow range of wavelengths.

This coherence makes laser diodes extremely useful in a wide range of applications, including telecommunications, optical data storage, laser printing, laser pointers, barcode scanners, medical devices, and more.

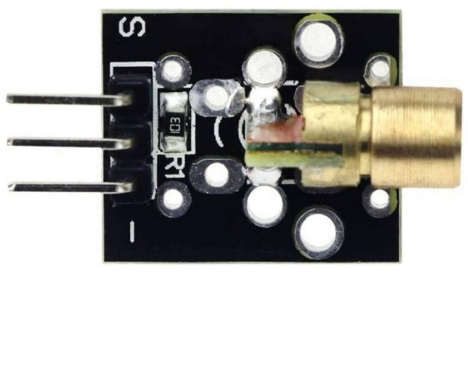


Figure 2.4: Laser Diode.

The component used in this work is a single 650nm 5mW LD, with an operational input voltage of 3 V, which nicely aligns with the GPIO output one. Moreover, in order to limit the current passing through the diode, 330 Ω resistor is configured in series with it.

2.4. Beam Splitter

A beam splitter is an optical device used to divide or split a beam of light into two or more separate beams. It operates by allowing a portion of the incident light to pass through (transmit) while reflecting another portion (reflect). Beam splitters are essential components in various optical systems and experiments, enabling functions such as beam combination, interference, polarization control, and signal routing. They come in different types, designs, and configurations to suit specific applications and requirements.

Specifically, the type of prism used in the optical section of the QRBG is a 50/50 beam splitter, which equally divides light based on its polarization state, rather than its intensity. It consists of a birefringent crystal or a stack of multiple layers with different refractive indices, arranged to split s-polarized and p-polarized light into separate paths.

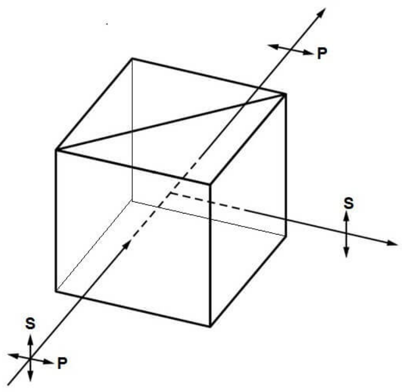


Figure 2.5: Polarizing 50/50 Beam Splitter working mechanism.

The key feature of this component is its ability to split the incident beam of light uniformly in the two fundamental polarization direction. It is this characteristics that corresponds to the implementation of the Hadamard gate for creating an equal superposition of observable states.

2.5. Light-Dependent Resistors

A light-dependent resistor (LDR), also known as a photoresistor or photocell, is a type of passive electronic component whose electrical resistance varies with the intensity of incident light. Essentially, it is a semiconductor device that exhibits a change in resistance when exposed to light.

An LDR is typically made from a semiconductor material with high resistance, such as cadmium sulfide (CdS) or cadmium selenide (CdSe). These materials have a property known as photoconductivity, where their conductivity increases when exposed to light.

In darkness or low-light conditions, the semiconductor material's resistance is high because there are few free charge carriers (electrons or holes) available to conduct current. When light falls on the LDR, photons with sufficient energy can excite electrons from the valence band to the conduction band, creating electron-hole pairs and increasing the number of charge carriers. This increase in charge carriers reduces the material's resistance, allowing more current to flow through the device.

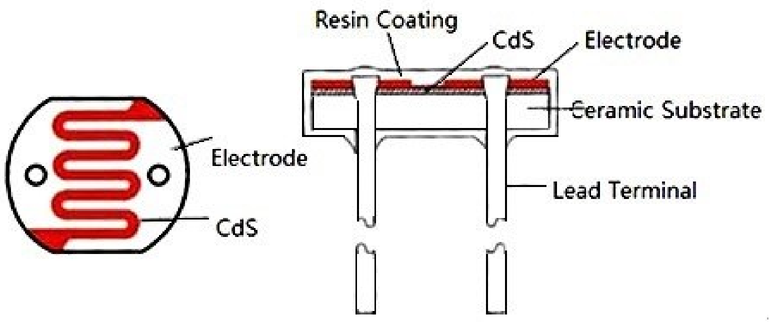


Figure 2.6: Light-Dependent Resistor layout.

With the supply being kept constant, and assuming a voltage divider configuration, a photoresistor is such that when light hits it lowering its resistance, more voltage falls across the LDR terminals.

It is this voltage that we read as an analog quantity to compare it between the two LDRs of the optical section of the QRBG architecture to ultimately determine which bit has been generated.

2.6. ADC

Being able to precisely read the voltage across the terminals of the system's photoresistors is a crucial aspect of the QRBG, as it is what ultimately determines the quality of how the theoretically perfect randomness is exploited. Voltage is, of course, an analog quantity, and while the Raspberry Pi is a device with lots of hardware/software functionalities indeed, one that it lacks is the integrated support for analog inputs.

On the other hand, some of the (digital) GPIOs do support multiple serial interfaces to communicate with external device, meaning that voltage readouts can still occur by incorporating an Analog-to-Digital Converter (ADC). This device serves two main purposes: it takes the burden of digitalizing the inputs on its analog channels, and manages the transfer of its conversions onto whichever devices asks it.

Here, the Adafruit ADS1115 ADC was chosen as a converter module. It is a precision, low-power, 16-bit, I^2-compatible module offered in an ultra-small, leadless, X2QFN-10 package, and a VSSOP-10 package. The ADS1115 devices incorporate a low-drift voltage reference and an oscillator.

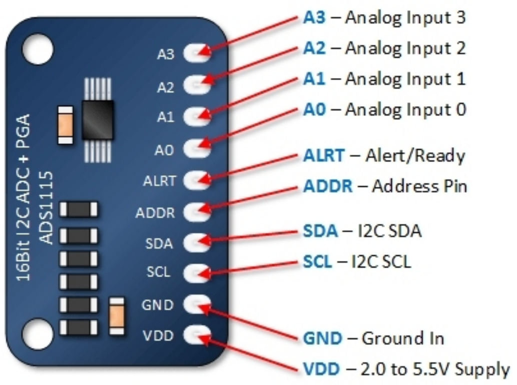


Figure 2.7: Adafruit ASD1115 Analog-to-Digital Converter.

This device also incorporates a programmable gain amplifier (PGA) and a digital comparator. These features, along with a wide operating supply range, make the ADS1115 well suited for power- and space-constrained, sensor measurement applications.

It performs conversions at data rates up to 860 samples per second (SPS). The PGA offers input ranges from ±256 mV to ±6.144 V, allowing precise large- and small-signal measurements. The ADS1115 features an input multiplexer (MUX) that allows two differential or four single-ended input measurements. The digital comparator is used for under- and overvoltage detection.

ADC core measures a differential signal, V_{IN}, that is the difference of $V_{(AINP)}$ and $V_{(AINN)}$. The converter core consists of a differential, switched-capacitor delta-sigma ($\Delta\Sigma$) modulator followed by a digital filter. This architecture results in a very strong attenuation of any common-mode signals. Input signals are compared to the internal voltage reference. The digital filter receives a high-speed bitstream from the modulator and outputs a code proportional to the input voltage. Its block diagram is depicted in Figure 2.8 .

The module has two available conversion modes: single-shot and continuous-conversion. In

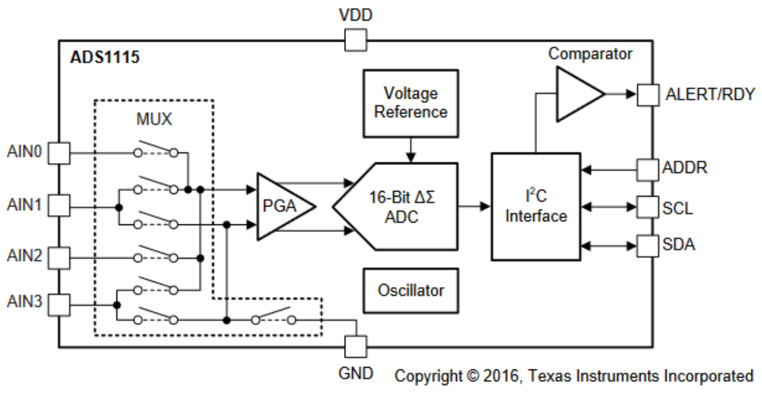
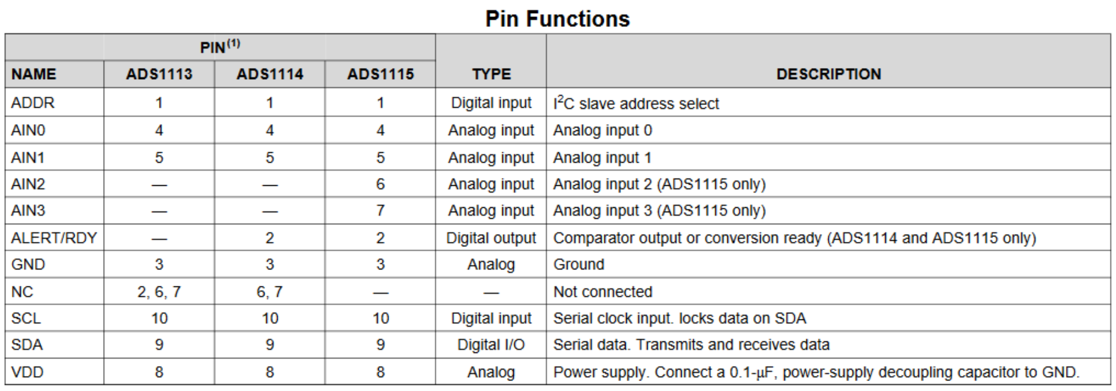


Figure 2.8: ADS1115 Block Diagram.

single-shot mode, the ADC performs one conversion of the input signal upon request, stores the conversion value to an internal conversion register, and then enters a power-down state. This mode is intended to provide significant power savings in systems that only require periodic conversions or when there are long idle periods between conversions.

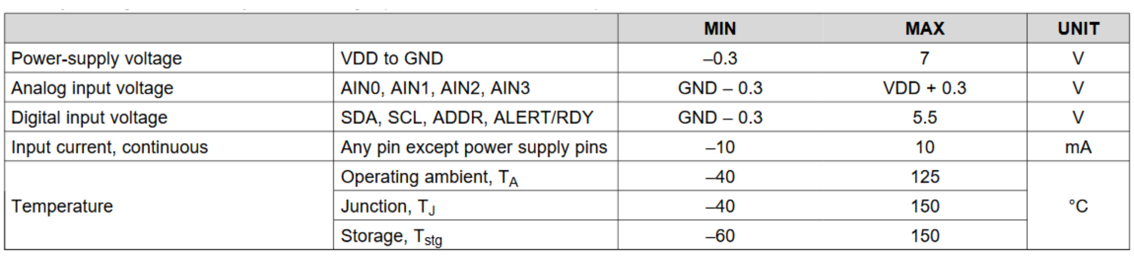
In continuous-conversion mode, the ADC automatically begins a conversion of the input signal as soon as the previous conversion is completed. The rate of continuous conversion is equal to the programmed data rate. Data can be read at any time and always reflect the most recent completed conversion.



Pin Functions

	PIN[1]				
NAME	ADS1113	ADS1114	ADS1115	TYPE	DESCRIPTION
ADDR	1	1	1	Digital input	I²C slave address select
AIN0	4	4	4	Analog input	Analog input 0
AIN1	5	5	5	Analog input	Analog input 1
AIN2	—	—	6	Analog input	Analog input 2 (ADS1115 only)
AIN3	—	—	7	Analog input	Analog input 3 (ADS1115 only)
ALERT/RDY	—	2	2	Digital output	Comparator output or conversion ready (ADS1114 and ADS1115 only)
GND	3	3	3	Analog	Ground
NC	2, 6, 7	6, 7	—	—	Not connected
SCL	10	10	10	Digital input	Serial clock input. locks data on SDA
SDA	9	9	9	Digital I/O	Serial data. Transmits and receives data
VDD	8	8	8	Analog	Power supply. Connect a 0.1-μF, power-supply decoupling capacitor to GND.

Figure 2.9: ADS111X ADC Family Pin Functions.



		MIN	MAX	UNIT
Power-supply voltage	VDD to GND	–0.3	7	V
Analog input voltage	AIN0, AIN1, AIN2, AIN3	GND – 0.3	VDD + 0.3	V
Digital input voltage	SDA, SCL, ADDR, ALERT/RDY	GND – 0.3	5.5	V
Input current, continuous	Any pin except power supply pins	–10	10	mA
Temperature	Operating ambient, T_A	–40	125	°C
	Junction, T_J	–40	150	
	Storage, T_{stg}	–60	150	

Figure 2.10: Operating condition of ADS1115.

3
Configuration

In this chapter, both the hardware layout, as well as the software configuration of the entire system is presented, starting from a more general overview of the project, subsequently delving into its details.

3.1. General layout

The components introduced in the previous section are placed in the physical configuration displayed in Figure 3.1, while the circuit connections are explicited in Figure 3.2. The flow of information starts at the bottom of the picture and proceeds counter-clockwise through it:

1. the controlling Forth code is first uploaded to the main controller through the CP2102 module with the UART protocol;
2. the Raspberry GPIOs are connected to the breadboard through a T-Cobbler extension, providing both power supply and data interface to the rest of the circuit;
3. as a first operation, the Laser is activated through GPIO Pin 11 configured in output mode and its supply is regulated by a 330 Ω resistor;
4. the generated beam of photons incides onto the 50/50 Beam Splitter which divides it into two main direction, a straight and a 90 degrees one;
5. the two separated beams land onto a photoresistor each; those LDRs are configured in a voltage divider with a 10 $k\Omega$ resistance, and the middle node is directly connected to one of the analog inputs on the ADS1115 ADC;
6. as a last step, the controller makes sure that an updated digital conversion value is ready and then initiates an I^{2}C read operation from the ADC slave device, subsequently post-processing the received value to interpret it as a generated bit.

3.2. Laser Interface

Managing the Laser is a task that reduces to a simple modification of the GPFSEL, GPSET, and GPCLR registers of the BCM2711. Specifically, the LD supply is connected to GPIO Pin 11, whose mode of operation, according to the peripherals reference, is handled by bits 5:3 of GPFSEL1, whose 0b001 config yields the desired output mode setup.

With the same simplicity it is also possible to configure the LD on/off action, through setting bit 11 of registers GPSET0 and GPCLR0 respectively. The SET operation provides the pin with the desired 3.3 V, while CLR defaults it back to GND, effectively turning the Laser on and off.

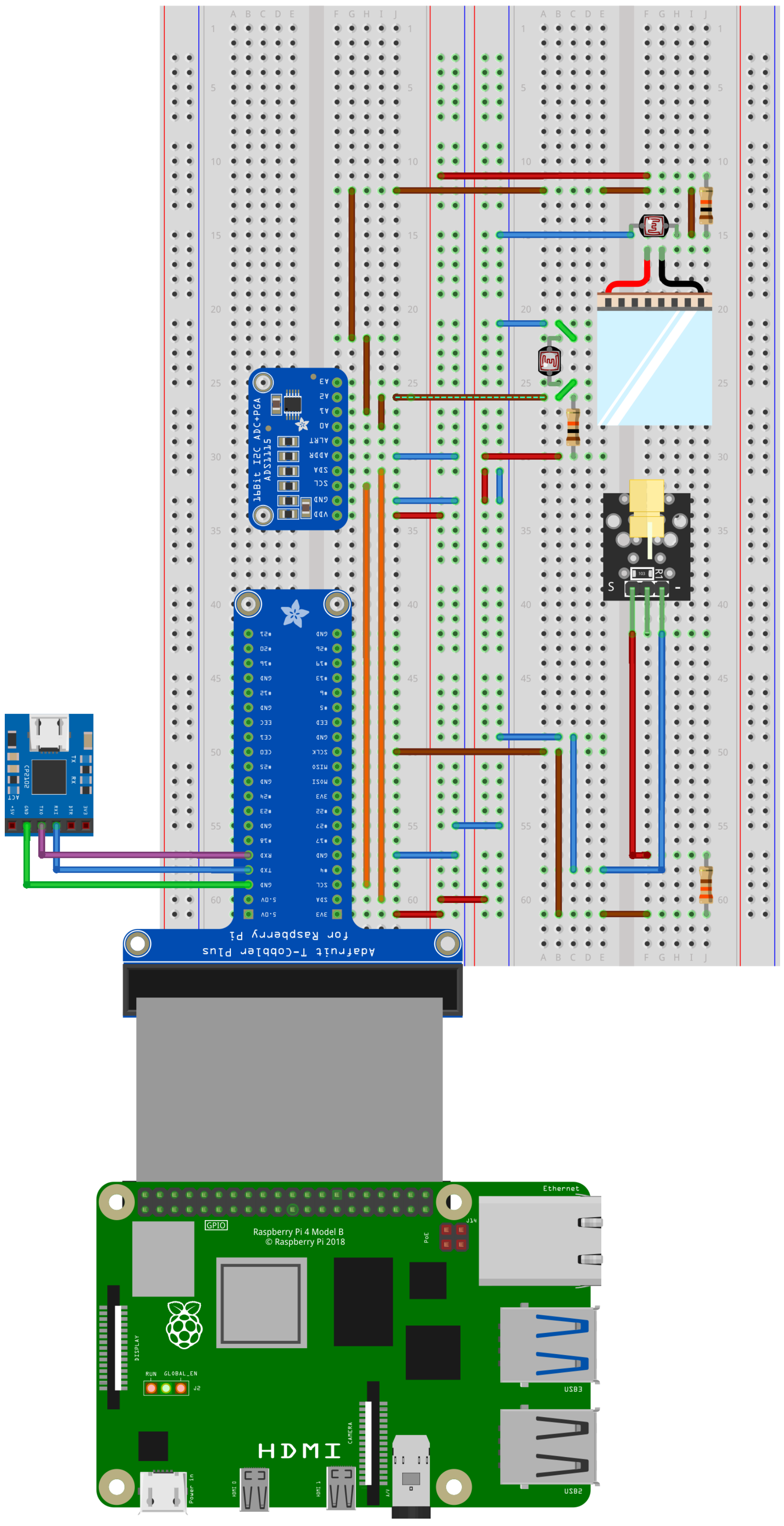


Figure 3.1: Hardware layout of the QRBG.

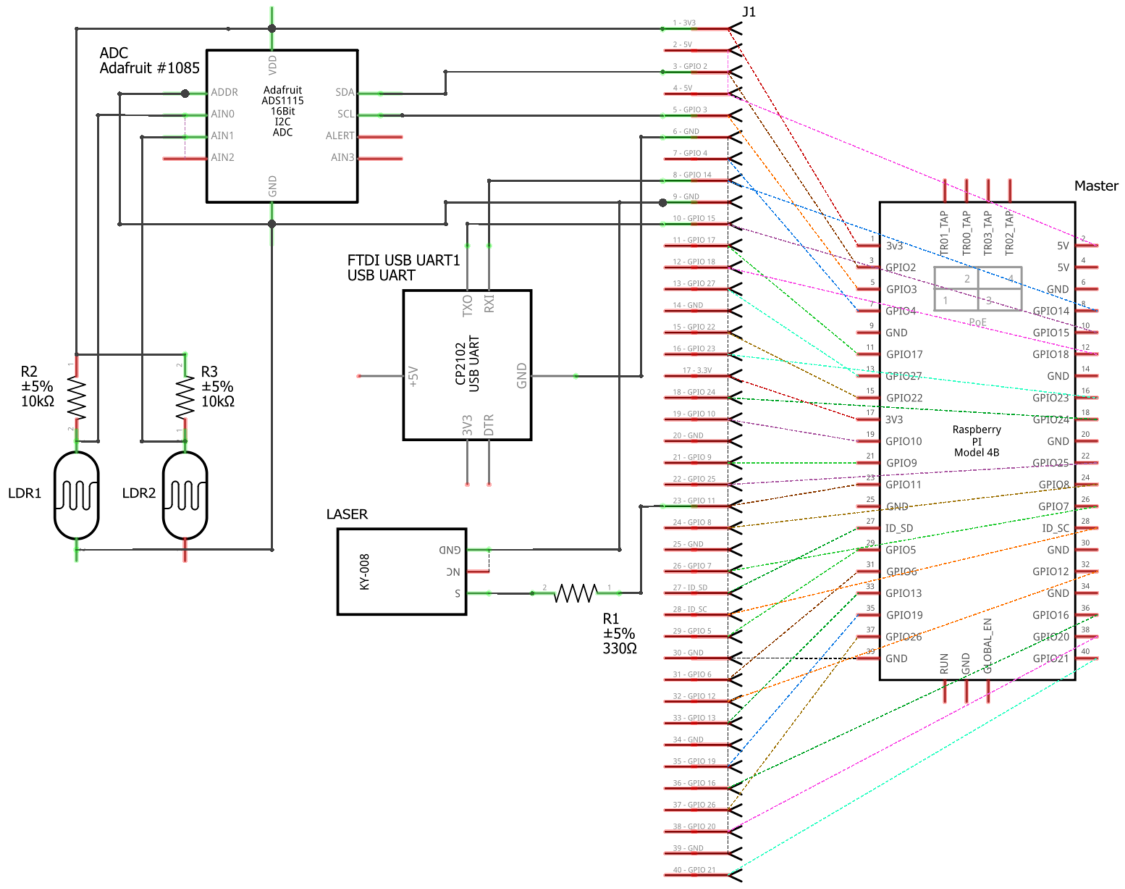


Figure 3.2: Internal connections of the QRBG.

3.3. LDR Config

In order to have a controlled terminal section whose voltage can be reliably measured to visualize the effect of light onto a photoresistor, a voltage divider configuration is required (Figure 2.6).

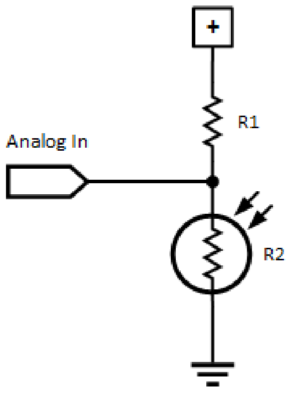


Figure 3.3: Voltage Divider configuration for LDR analog readout.

A voltage divider is a simple circuit configuration used to generate a voltage output that is a fraction of the input voltage. It's commonly employed in various electronic circuits for tasks like level shifting, setting reference voltages, and in this case, reading analog signals such as those from a photoresistor.

In our case, with VDD at supply and V_{AIN} the analog input connected from the divider to the ADC, the output voltage is computed according to

$$V_{AIN} = VDD\,\frac{R1}{R1 + LDR}$$

When the resistance of the photoresistor changes with the light intensity falling on it, as the resistance changes, it alters the voltage division ratio between $R1$ and LDR, resulting in a change in V_{AIN}.

Since two of those configurations are present in the overall architecture, handling both of them may be done in two ways: first, each V_{AIN} connected to a different input channel on the ADC is read separately with respect to GND, and the two conversions are compared in order to determine which bit is generated; second, the ADC is configured so that a conversion is made on the voltage across two different input channels, rather than just with respect to ground, which is indeed the chosen setup, as by operating this way post-processing is ever more immediate and, most importantly, just one conversion is needed rather than two, effectively doubling the speed of the entire system.

3.4. I^{2}C Readouts

3.4.1. Hardware Setup

The ADS1115 ADC device used for analog readouts is capable of interfacing with external devices through the I^{2}C protocol. Layout-wise, it is important that a final configuration looks something like that of Figure 3.4. There, the following connections need to be taken special care of:

- **SCL** and **SDA**: the two main bus interfaces (Serial Clock and Serial Data) must be connected to the appropriate I^{2}C-compatible GPIOs on the master device, which in the Raspberry corresponds to Pins 2 and 3, which are managed by the GPIO Alternative Function 0 (ALT0). This alternative is selected by configuring the corresponding bits on the BCM2711 GPFSEL register.
- **Pull-up resistance**: the Open-Drain configuration on which the I^{2}C is based on requires pull-up resistors on both clock and data lines shared by masters and slaves. Luckily enough, no setup is needed here as the GPIOs of the Raspberry Pi have integrated 1.8 $k\Omega$ pull-up resistors into them.
- **Slave Address**: in order to initiate a communication, the I^{2}C master needs to identify the slave device. This is done by having the master send on the SDA line the hardware address of the slave, with the latter sending an ACK bit to confirm the reception of contact. The ADS1115 is such that its slave address is configurable, with the default value corresponding to 48h, which is set by grounding the ADDR pin on the module, as in our case.

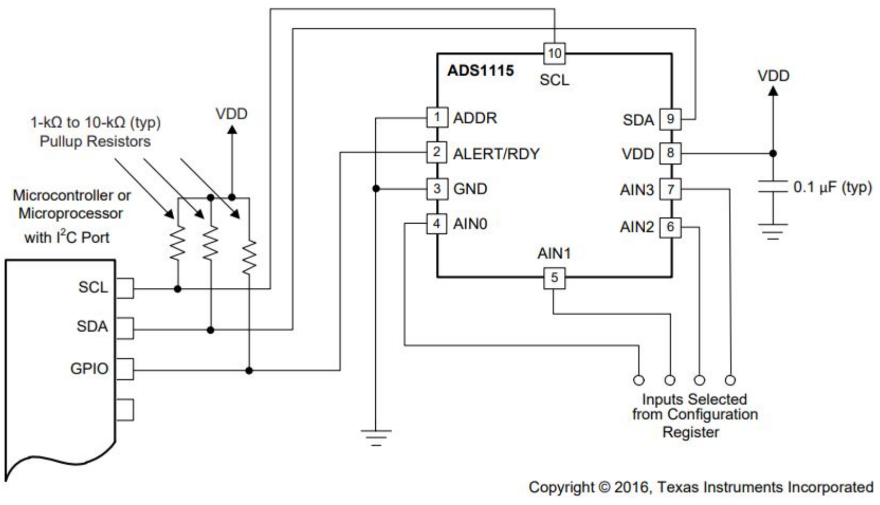
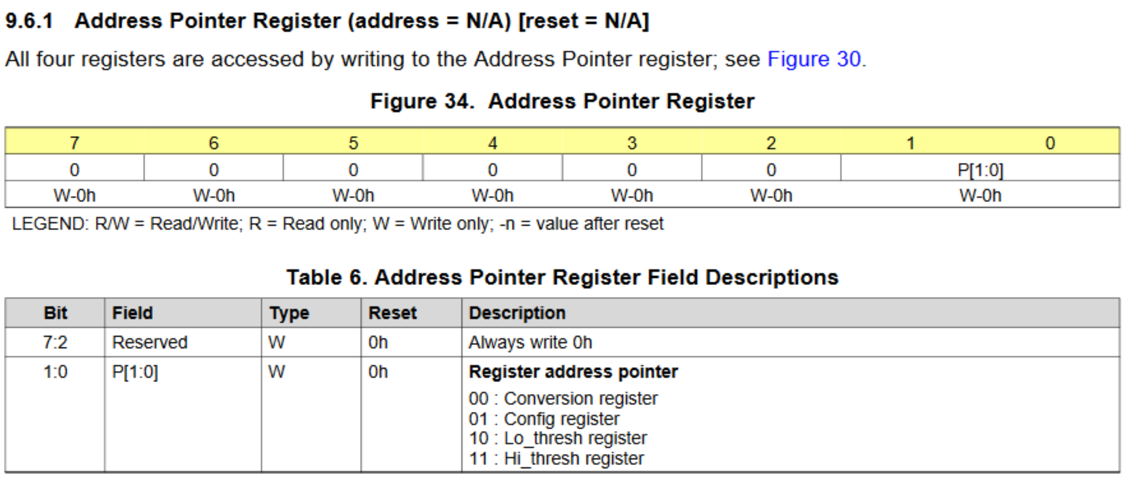


Figure 3.4: Connections of the ADS1115.

Additional ADC pins such as ALERT/RDY or unused analog inputs can instead be safely left floating.

3.4.2. ADS1115 Registers

The ADS1115 has four registers that are accessible through the I2C interface using the **Address Pointer register** (Figure 3.5). The **Conversion register** contains the result of the last conversion. The **Config register** is used to change the ADS1155 operating modes and query the status of the device. The other two registers, Lo_thresh and Hi_thresh, set the threshold values used for the comparator function.



9.6.1 Address Pointer Register (address = N/A) [reset = N/A]

All four registers are accessed by writing to the Address Pointer register; see Figure 30.

Figure 34. Address Pointer Register

7	6	5	4	3	2	1	0
0	0	0	0	0	0	P[1:0]	
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	

LEGEND: R/W = Read/Write; R = Read only; W = Write only; -n = value after reset

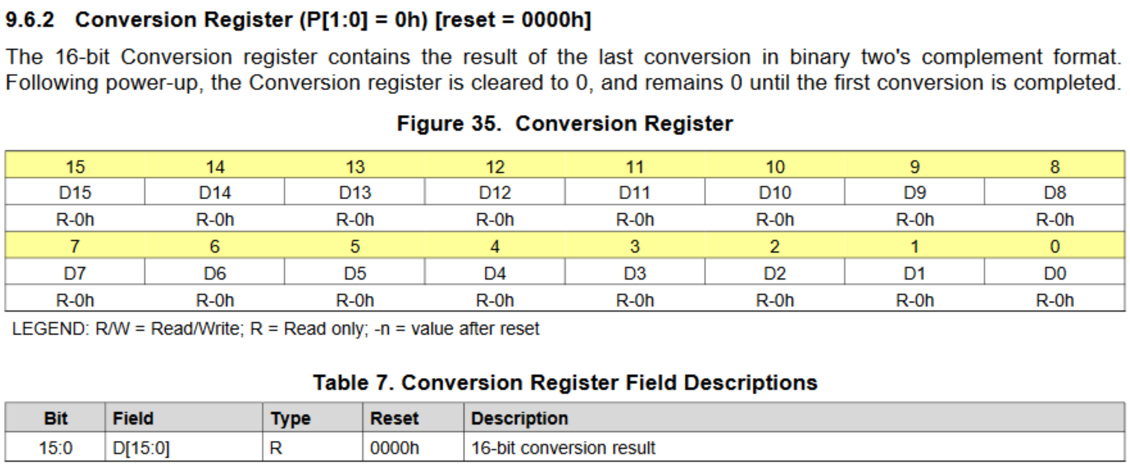
Table 6. Address Pointer Register Field Descriptions

Bit	Field	Type	Reset	Description
7:2	Reserved	W	0h	Always write 0h
1:0	P[1:0]	W	0h	**Register address pointer** 00 : Conversion register 01 : Config register 10 : Lo_thresh register 11 : Hi_thresh register

Figure 3.5

Writing to the Address Pointer to this register that selects which other registers to read from in subsequent stages of the communication: after the slave address is sent and acknowledged, in the case of a read operation, the master takes care of sending as a second byte the content of Address Pointer to tell the slave where to attach the following read operation.

The Conversion register, on the other hand, is obviously just read-only, as it is used to store the 16-bit precision result of the analog-to-digital conversion performed by the module. It is the value contained in this register that is sent to the master through a read operation and that is used to determine which bit is being generated.



9.6.2 Conversion Register (P[1:0] = 0h) [reset = 0000h]

The 16-bit Conversion register contains the result of the last conversion in binary two's complement format. Following power-up, the Conversion register is cleared to 0, and remains 0 until the first conversion is completed.

Figure 35. Conversion Register

15	14	13	12	11	10	9	8
D15	D14	D13	D12	D11	D10	D9	D8
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
D7	D6	D5	D4	D3	D2	D1	D0
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

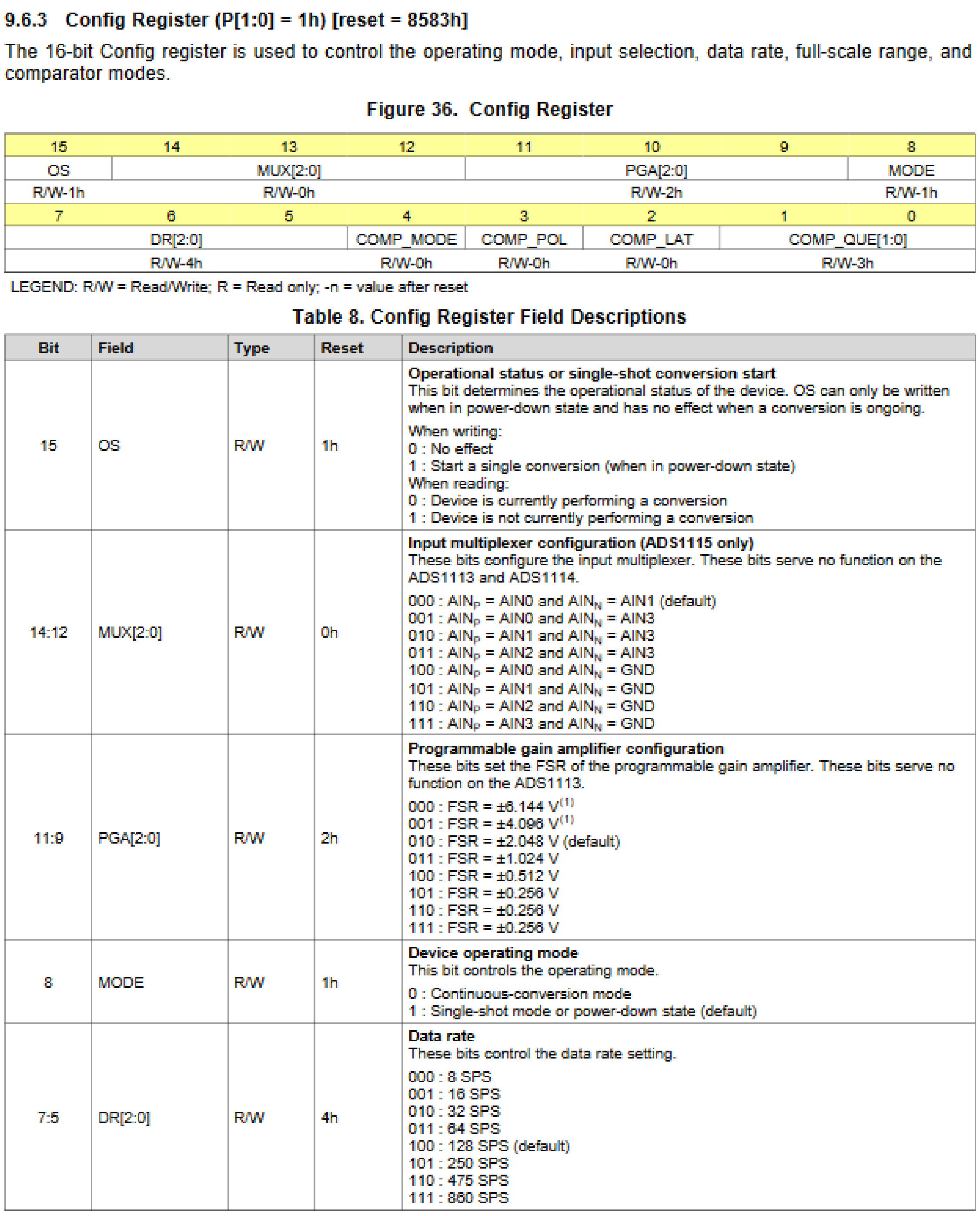
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 7. Conversion Register Field Descriptions

Bit	Field	Type	Reset	Description
15:0	D[15:0]	R	0000h	16-bit conversion result

Figure 3.6

The last register of use for this study case is the Config register. This register is also 16-bit and contains the setup configuration for the operational modes of the ADS1115. Other than the setup of initial parameters, write to this register are required, when in single-shot mode, to trigger a conversion.



9.6.3 Config Register (P[1:0] = 1h) [reset = 8583h]

The 16-bit Config register is used to control the operating mode, input selection, data rate, full-scale range, and comparator modes.

Figure 36. Config Register

15	14	13	12	11	10	9	8
OS	MUX[2:0]			PGA[2:0]			MODE
R/W-1h	R/W-0h			R/W-2h			R/W-1h
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DR[2:0]			COMP_MODE	COMP_POL	COMP_LAT	COMP_QUE[1:0]	
R/W-4h			R/W-0h	R/W-0h	R/W-0h	R/W-3h	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 8. Config Register Field Descriptions

Bit	Field	Type	Reset	Description
15	OS	R/W	1h	**Operational status or single-shot conversion start** This bit determines the operational status of the device. OS can only be written when in power-down state and has no effect when a conversion is ongoing. When writing: 0 : No effect 1 : Start a single conversion (when in power-down state) When reading: 0 : Device is currently performing a conversion 1 : Device is not currently performing a conversion
14:12	MUX[2:0]	R/W	0h	**Input multiplexer configuration (ADS1115 only)** These bits configure the input multiplexer. These bits serve no function on the ADS1113 and ADS1114. 000 : AIN$_P$ = AIN0 and AIN$_N$ = AIN1 (default) 001 : AIN$_P$ = AIN0 and AIN$_N$ = AIN3 010 : AIN$_P$ = AIN1 and AIN$_N$ = AIN3 011 : AIN$_P$ = AIN2 and AIN$_N$ = AIN3 100 : AIN$_P$ = AIN0 and AIN$_N$ = GND 101 : AIN$_P$ = AIN1 and AIN$_N$ = GND 110 : AIN$_P$ = AIN2 and AIN$_N$ = GND 111 : AIN$_P$ = AIN3 and AIN$_N$ = GND
11:9	PGA[2:0]	R/W	2h	**Programmable gain amplifier configuration** These bits set the FSR of the programmable gain amplifier. These bits serve no function on the ADS1113. 000 : FSR = ±6.144 V(1) 001 : FSR = ±4.096 V(1) 010 : FSR = ±2.048 V (default) 011 : FSR = ±1.024 V 100 : FSR = ±0.512 V 101 : FSR = ±0.256 V 110 : FSR = ±0.256 V 111 : FSR = ±0.256 V
8	MODE	R/W	1h	**Device operating mode** This bit controls the operating mode. 0 : Continuous-conversion mode 1 : Single-shot mode or power-down state (default)
7:5	DR[2:0]	R/W	4h	**Data rate** These bits control the data rate setting. 000 : 8 SPS 001 : 16 SPS 010 : 32 SPS 011 : 64 SPS 100 : 128 SPS (default) 101 : 250 SPS 110 : 475 SPS 111 : 860 SPS

Figure 3.7

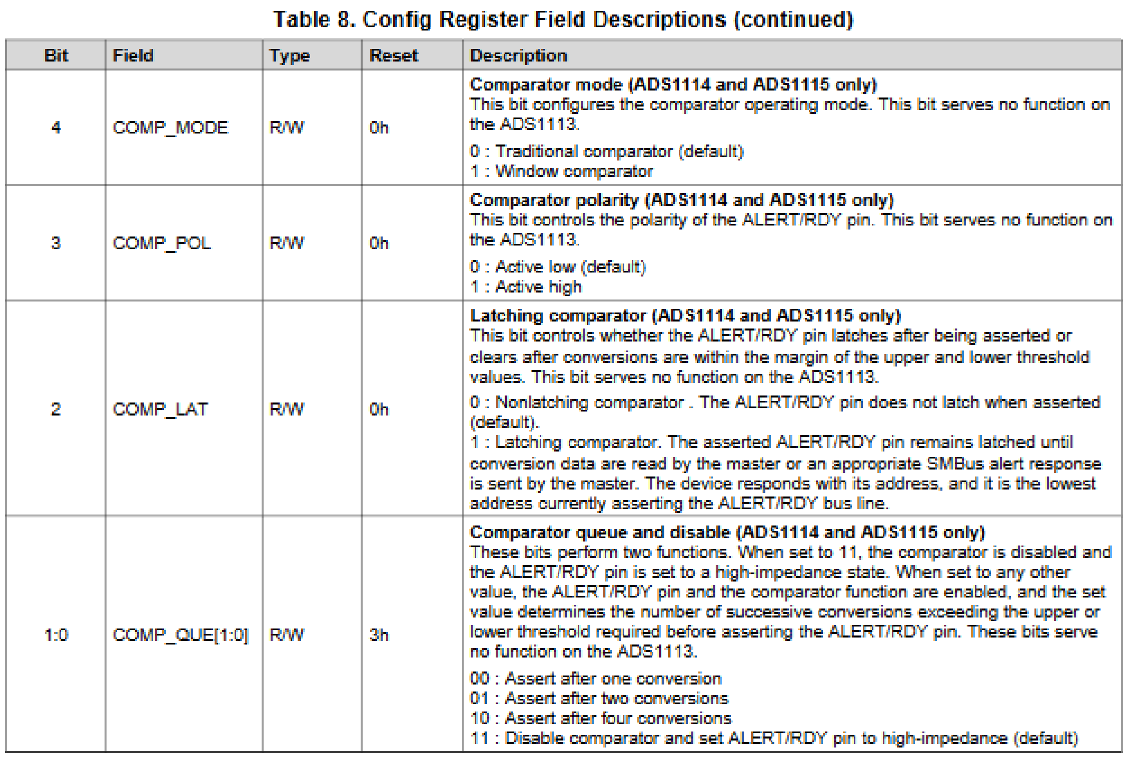


Table 8. Config Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
4	COMP_MODE	R/W	0h	**Comparator mode (ADS1114 and ADS1115 only)** This bit configures the comparator operating mode. This bit serves no function on the ADS1113. 0 : Traditional comparator (default) 1 : Window comparator
3	COMP_POL	R/W	0h	**Comparator polarity (ADS1114 and ADS1115 only)** This bit controls the polarity of the ALERT/RDY pin. This bit serves no function on the ADS1113. 0 : Active low (default) 1 : Active high
2	COMP_LAT	R/W	0h	**Latching comparator (ADS1114 and ADS1115 only)** This bit controls whether the ALERT/RDY pin latches after being asserted or clears after conversions are within the margin of the upper and lower threshold values. This bit serves no function on the ADS1113. 0 : Nonlatching comparator . The ALERT/RDY pin does not latch when asserted (default). 1 : Latching comparator. The asserted ALERT/RDY pin remains latched until conversion data are read by the master or an appropriate SMBus alert response is sent by the master. The device responds with its address, and it is the lowest address currently asserting the ALERT/RDY bus line.
1:0	COMP_QUE[1:0]	R/W	3h	**Comparator queue and disable (ADS1114 and ADS1115 only)** These bits perform two functions. When set to 11, the comparator is disabled and the ALERT/RDY pin is set to a high-impedance state. When set to any other value, the ALERT/RDY pin and the comparator function are enabled, and the set value determines the number of successive conversions exceeding the upper or lower threshold required before asserting the ALERT/RDY pin. These bits serve no function on the ADS1113. 00 : Assert after one conversion 01 : Assert after two conversions 10 : Assert after four conversions 11 : Disable comparator and set ALERT/RDY pin to high-impedance (default)

Figure 3.8

3.4.3. Read/Write Operations

With the connections laid out, the system becomes ready for communication. Data exchange between the master and the ADC registers occurs according to the standard byte-wise I^{2}C protocol, which can be implemented on the master through its Broadcom Serial Control (BSC) controller of the BCM2711 chip.

Writing operations occur with the following procedure, depicted also in Figure 3.9:

1. the master initiates communication by triggering an I^{2}C Start Condition, corresponding to a transition of the SDA line from an Idle state (logic high level) to an Active state (logic low level) while the SCL line is idle;
2. then it starts writing on the bus the 7-bit address, along with an 8th WRITE bit, to which the slave responds with an ACK bit;
3. the master follows up by writing the address of the slave's register that is going to be updated, which the ADS1115 puts in its Address Point Register and again ACKs to the master;
4. eventually, the third byte sent by the master is the actual most significant one of the data that is sent to the slave; the communication continues byte-by-byte until the least significant one, followed by a Stop Condition by the master.

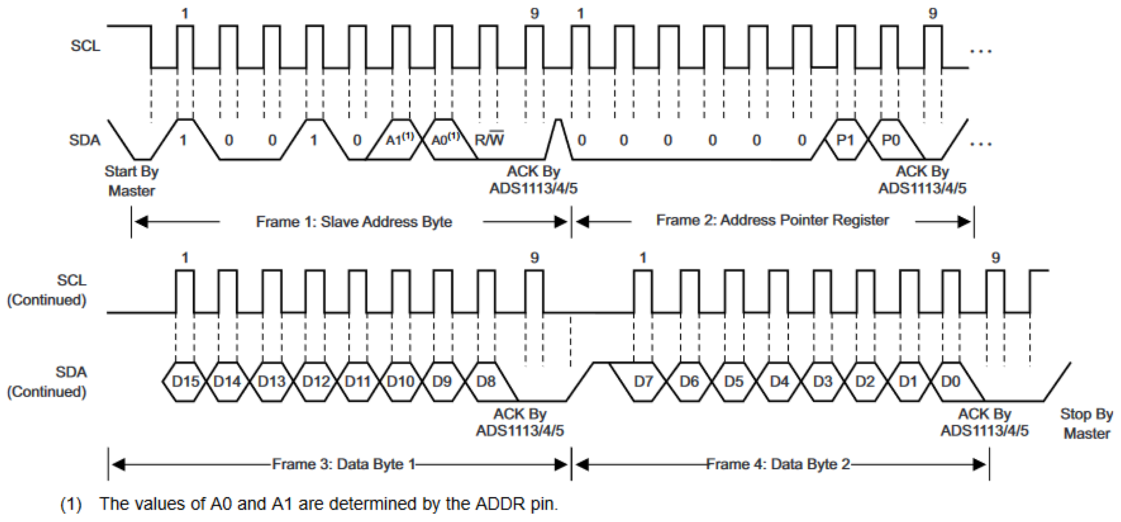


Figure 3.9: Timing diagram from writing to ADS1115.

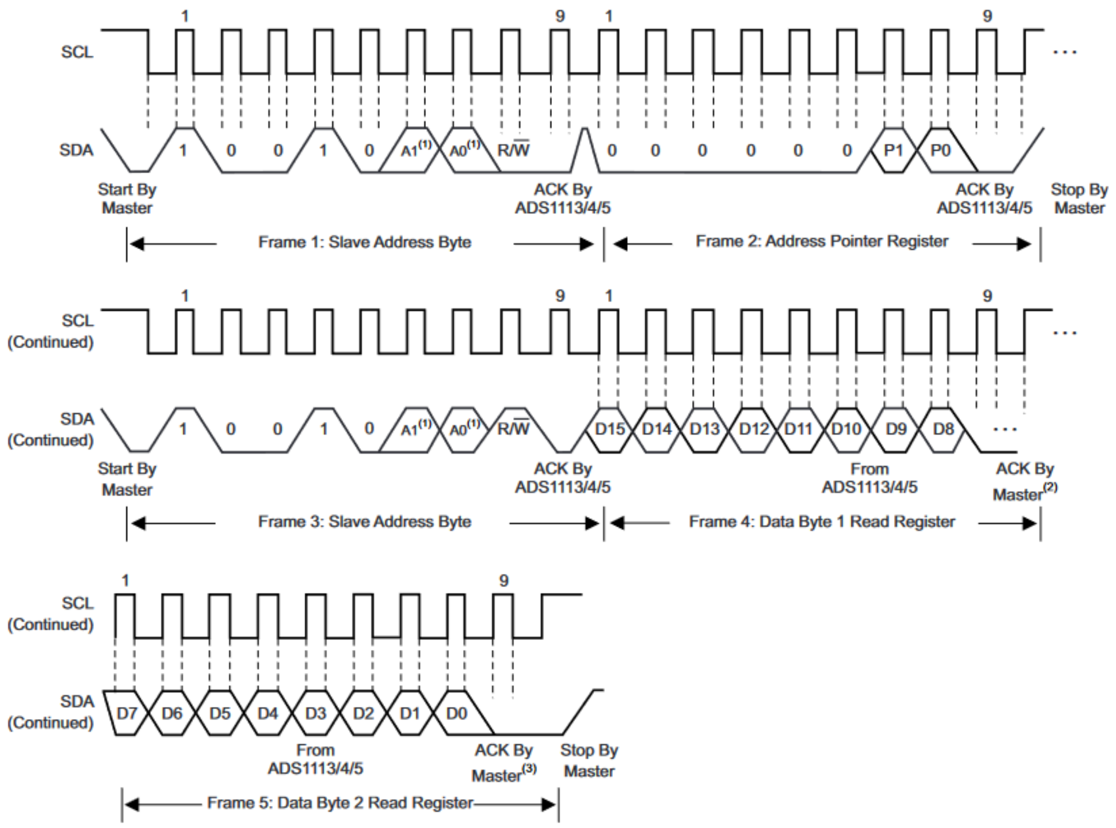
In this context, write operations are employed in two occasions:

- when a new ADC configuration has to be set, including the case when, single-shot mode, requires updating the most significant bit of the Configuration Register;
- before a read operation, since the reading mechanism involves a preliminary write to the Address Pointer Register of the address that is going to be read.

Read operations, differently from write ones, are slightly more complex and are performed as follows:

1. just as before, a to-be-ACKed slave address + WRITE bit are written on the bus after the Start Condition;
2. as a second byte, the address of the slave's device to read from is specified;
3. then, the master keeps the connection alive by sending a Repeated Start Condition (logically equal to the Start one) with a READ bit set;
4. the slave acknowledges the operation and starts bit-banging the content of the specified register to the bus, one byte at a time;

5. the master will then use its internal logic to handle the received data, i.e., by putting it in a buffer waiting to be extracted.



(1) The values of A0 and A1 are determined by the ADDR pin.
(2) Master can leave SDA high to terminate a single-byte read operation.
(3) Master can leave SDA high to terminate a two-byte read operation.

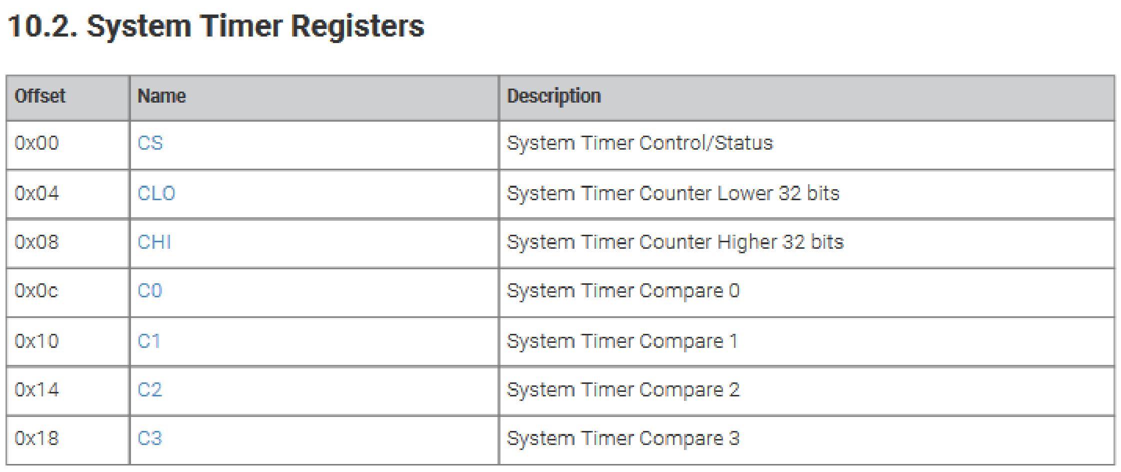
Figure 3.10: Timing daigram from reading to ADS1115.

3.5. Synchronization

Naturally, in an ideal scenario, the Random Bit Generation can occur at really high speeds, enhancing the performance of its communication security applications. While the model developed in this work is theoretically optimal in terms of computational time, in the sense that it is just as fast as the main controller processor, in practice, the whole system has one significant bottleneck: the sample rate of the ADC. The chosen ADS1115 module, despite being a pretty good working device, especially in relation to its cost, is definitely not the fastest one available, with a top 860 SPS speed, which of course is much lower than the 1.5 GHz clock speed of the main controller.

For this reason, being that, in the end, the overall bit generation is indeed paced by the ADC, there is no need of continuously polling the ADC for new conversion results when we are sure that a new one has not yet occurred. In the resource-efficiency-oriented philosophy of Embedded Systems, unnecessary processor load can be relieved if ADC-read actions are appropriately timed.

This is achieved by exploiting the System Timer of the BCM2711.



10.2. System Timer Registers

Offset	Name	Description
0x00	CS	System Timer Control/Status
0x04	CLO	System Timer Counter Lower 32 bits
0x08	CHI	System Timer Counter Higher 32 bits
0x0c	C0	System Timer Compare 0
0x10	C1	System Timer Compare 1
0x14	C2	System Timer Compare 2
0x18	C3	System Timer Compare 3

Figure 3.11: System Timer register of the BCM2711.

The System Timer peripheral provides four 32-bit timer channels and a single 64-bit, 1MHz (fixed), free running counter. Each channel has an output compare register, which is compared against the 32 least significant bits of the free running counter values. When the two values match, the system timer peripheral generates a signal to indicate a match for the appropriate channel. The match signal is then fed into the interrupt controller. The interrupt service routine then reads the output compare register and adds the appropriate offset for the next timer tick. The free running counter is driven by the timer clock and stopped whenever the processor is stopped in debug mode.

As a consequence, optimal mode of operation may consist in regulating the timer according to a fixed 0x48B constant, which is the number obtained after rounding the division of the 1 MHz free-running counter frequency with the configured maximum sample-per-second rate of the ADC, equal to 860 SPS.

This translates to a more efficient system, since the main controller takes advantage of the timer to wait for the ADC to perform a new conversion, occupying the intra-conversion time with just the poll of the CS register of BCM2711, rather than entire I^{2}C exchanges.

4
Tests

4.1. Benchmark

Having defined the entire architecture, and having also assessed its computational performance, we are left with the evaluation of the quality of randomness of generated bits. Here, with the term "quality", we refer to the desired feature of uniformity of probability distribution, as well as unpredictability of sequences.

To benchmark those aspects, multiple runs of the bit generation process have been executed under a fixed hardware environment (i.e., as it can be imagined, a precise physical distance must be set between the Beam Splitter and the LDRs, in order to match the overall intensity of the light inciding on the sensors), then, the output bits dumped to the ZOC8 terminal were exported outside of the main development environment to a Python script to ease the plotting of the gathered data.

The first test, whose output is depicted in Figure 4.1 involves a the simple generation of a random bit stream to analyze the frequency of 0s and 1s, which we indeed observe being roughly 50/50 split between the two, as we would expect.

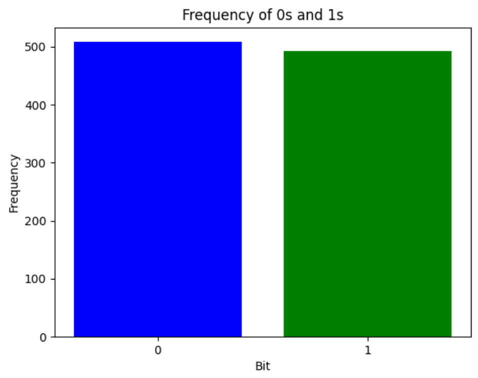


Figure 4.1: Frequency distribution of the generated 1000-bit stream.

As a second test, 1024 generated bits were grouped 8 at a time, and interpreted as a single-byte number. Then, just as before, the frequency of each number was plotted (Figure 4.2), again displaying a distribution somewhat alike to a uniform one.

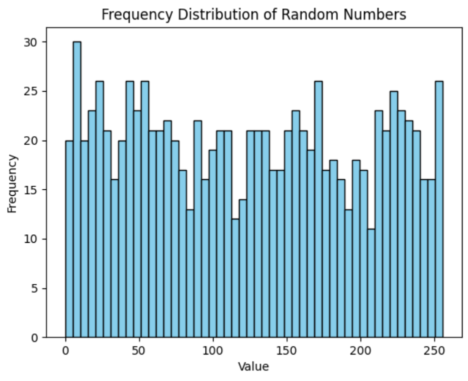


Figure 4.2: Frequency distribution of 8-bit numbers generated by the QRBG.

4.2. Conclusions

In conclusion, the development and implementation of the Quantum Random Bit Generator (QRBG) project represent an example of point of contact between the fields of embedded systems and quantum computing. By harnessing the principles of quantum mechanics, the QRBG offers a novel approach to random bit generation that surpasses traditional pseudo-random number generators (PRNGs) in terms of unpredictability, security, and reliability (at least in theory).

By intertwined design and optimization, a bare-metal QRBG capable of interfacing with embedded systems architectures was successfully created. By leveraging quantum phenomena such as photon detection or electronic noise, the QRBG produces truly random bits that are inherently unpredictable, meeting the stringent requirements of cryptographic applications, simulations, and secure communication protocols.

The QRBG project underscores the importance of interdisciplinary collaboration between quantum physics, electronics engineering, and computer science. It demonstrates the potential of quantum technologies to revolutionize conventional computing paradigms and address the growing demand for secure randomness in emerging technologies such as quantum computing and the Internet of Things (IoT).

Moving forward, further research and development efforts will focus on optimizing the QRBG for scalability, energy efficiency, and integration into diverse embedded systems platforms. Additionally, exploring the potential applications of quantum randomness in areas such as quantum cryptography, machine learning, and artificial intelligence will be essential for unlocking the full potential of this groundbreaking technology.