CLOSED LOOP ELECTRONIC SPEED CONTROLLER(ESC)

Schematic:

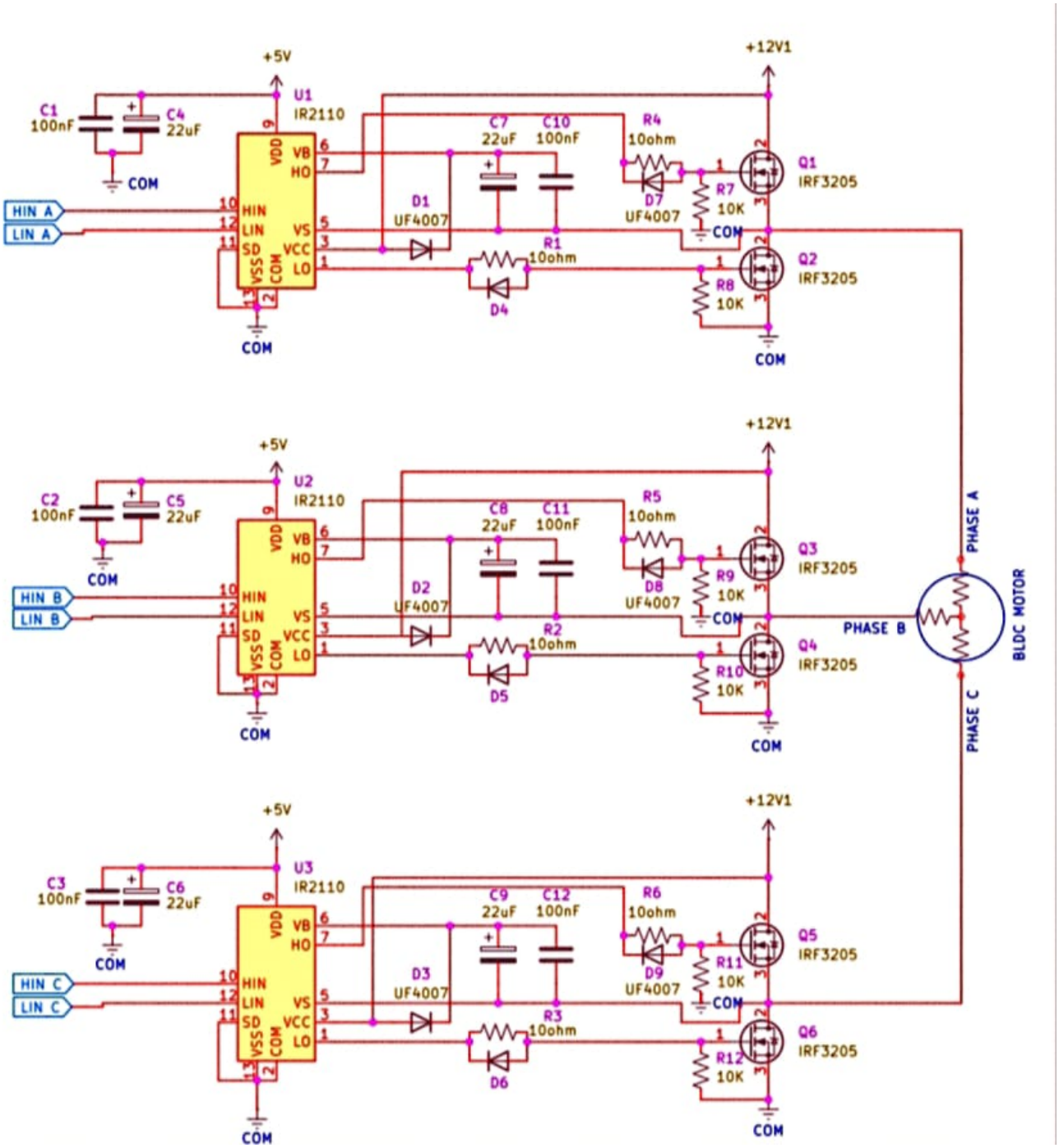

Figure 1: Circuit Diagram of Driver(Courtesy:CDOH,ICFOSS)

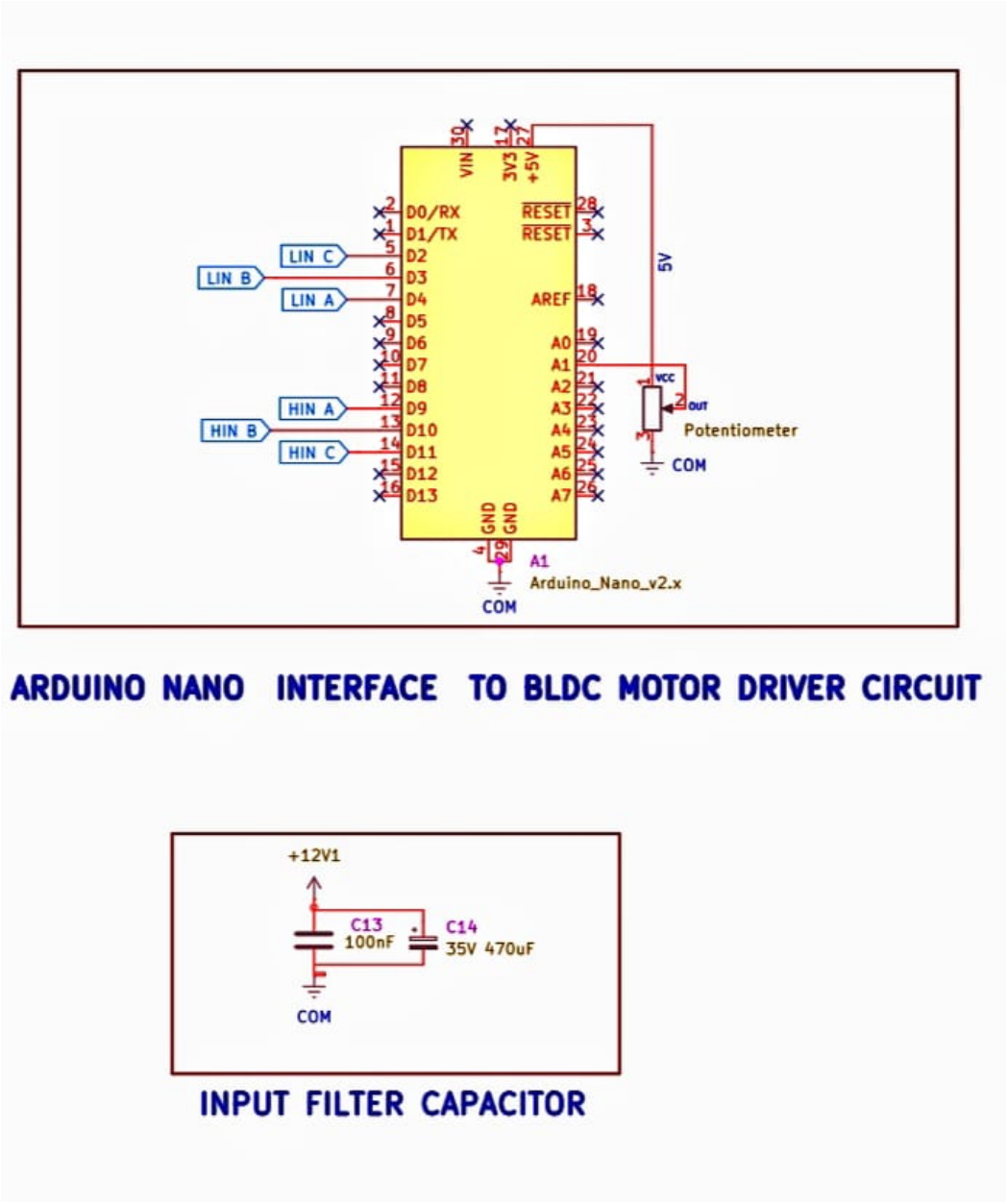

Figure 2: Hardware connections to Arduino Nano(Courtesy:CDOH,ICFOSS)

1 Implementation

1.1 Controlling motor in Open Loop Configuration using Timers

Timer Configuration

Timer1: Generates PWM signals for controlling one set of outputs (e.g., high-side drivers on specific pins such as digital 9 and 10).

Configuration: The timer is configured by setting its control registers (TCCR1A and TCCR1B) with no prescaling, meaning the system clock drives the timer directly. The settings (for example, setting TCCR1A to a value like 0x81) indicate that Timer1 operates in Phase Correct PWM mode with an 8-bit resolution. This mode produces a symmetric PWM waveform, which is beneficial for reducing switching noise and ensuring smooth motor operation.

Timer2: Provides an additional PWM output (e.g., for the remaining high-side driver on another pin such as digital 11) and helps synchronize commutation steps across all motor phases.

Configuration: Timer2 is similarly set up by configuring TCCR2A and TCCR2B with no prescaling and operates in Phase Correct PWM mode, ensuring a stable and symmetric PWM signal.

Role of Timers in Motor Commutation PWM Signal Generation: The timers generate PWM signals that drive the MOSFETs in the ESC circuit. These signals are essential for controlling the high-side switches in the H-bridge configuration used to energize the motor windings.

Commutation Timing: Fixed Commutation Sequence: Although the code uses an open-loop configuration (with fixed delays determined by the throttle input), the timers ensure that the PWM outputs remain stable during each commutation step. This stability allows for predictable switching of the motor phases along the 6 step commutation sequence

Duty Cycle Variation and Its Impact on Speed: (Throttle Signal Mapping) The PulseInput library captures a PWM signal from the transmitter (on a designated input pin). The pulse width of this signal(1008-2008) reflects the throttle position or desired speed. This mapping produces a value that is used to set the PWM duty cycle on the output compare registers of Timer1 and Timer2.A higher duty cycle means a higher average voltage, resulting in more power delivered to the motor and a higher rotational speed.

Register Updates: The duty cycle value is applied by updating the PWM registers (such as OCR1A, OCR1B, and OCR2A). These registers control the output signals on the specific pins connected to the motor drivers.

Code1:

GitHub - Open Loop prototype/ESC$_{t}imer_{o}pen_{l}oop_{a}rduino_{p}ulseinput_{l}ibrary.ino$

Results: Implemented open loop control using Timer1 and Timer2 in Phase Correct PWM mode and Transmitter control using PulseInput library and setting up delayMicroseconds() function between Commutation steps.

Figure 3: Output waveform of Open loop Commutation

Figure 4: PWM duty cycle Variation from the the PWM pins

1.2 Use of Pin Change Interrupt to adjust Throttle PWM value

Real-Time Throttle Mapping: Since the ISR continuously updates the PWM pulse width (using the pulseInput library in the previous program 1 or direct pin change in Program 2), the control algorithm always has the latest throttle value. This up-to-date information is then used to adjust both the PWM duty cycle and the commutation timing, ensuring that speed control is responsive to user input.

Improved Responsiveness and Flexibility: By avoiding blocking delays, the system can perform additional tasks concurrently (like sensor readings, safety checks, or further control computations). This flexibility is crucial in a closed-loop control system where continuous feedback and timely adjustments lead to smoother and more accurate speed control.

Enabling Closed-Loop Control: The interrupt-driven approach lays the groundwork for closed-loop control. It frees the main loop to quickly compare the desired motor speed (from the measured throttle) against the actual motor behavior and make real-time adjustments.

Code 2:

Github - Open Loop prototype/ESC$_{t}imer_{o}pen_{l}oop_{a}rduino_{p}inchangeinterrupt.ino$

Results: Building the motor control in open loop configuration and mapping the Transmitter throttle PWM input using a PinChange Interrupt to the PWM duty cycle assigned for motor commutation. Removing the delayMicroseconds() function helps in the reduction of blocking delays and enables in laying the groundwork for implementing a Closed loop system.

1.3 Implementation of Closed Loop

- **Hardware and Peripheral Initialization:** The microcontroller is configured to drive both high- and low-side switches of a three-phase H-bridge. Two timers are set up in PWM mode without prescaling to generate stable, high-frequency PWM signals. A pin change interrupt on the throttle input captures the PWM signal from the RC transmitter, and the analog comparator is configured for back-EMF sensing (with its interrupt initially disabled).
- **Throttle Signal Acquisition:** A pin change interrupt accurately captures the pulse width of the throttle signal by recording the time at rising and falling edges. This pulse width, which represents the user's throttle command, is stored for later use, eliminating the need for polling and ensuring a responsive measurement.
- **Mapping and Adjusting PWM Outputs:** In the main loop, the measured pulse width is first

constrained within expected limits. It is then mapped to two key control parameters: a PWM duty cycle (adjusting the average voltage and thus the motor speed) and a commutation interval (with a higher throttle reducing the interval for faster commutation).

- **Open-Loop Startup Routine:** Before engaging closed-loop feedback, the motor is spun up using an open-loop routine. This routine cycles through the fixed six-step commutation sequence with gradually decreasing delays until a minimum operational speed is reached, ensuring sufficient back-EMF is generated for reliable feedback.
- **Transition to Closed-Loop Operation:** Once the motor achieves the required speed, the system transitions to closed-loop mode by enabling the analog comparator interrupt. The comparator then monitors the back-EMF on the non-energized phase to provide real-time rotor position feedback.
- **Closed-Loop Commutation:** In closed-loop mode, the comparator's ISR is triggered upon detecting a valid zero-crossing, with noise filtering to ensure reliability. This feedback advances the six-step commutation sequence, with each step properly setting the output pins and configuring the timers to drive the H-bridge, thereby ensuring precise and efficient motor control.
- **Overall Operation Summary:** The system measures the throttle input via interrupts and maps it to both PWM duty cycle and commutation timing. An initial open-loop startup spins the motor, after which closed-loop control takes over, using back-EMF feedback to synchronize commutation with the rotor's position. This combined approach enhances efficiency, responsiveness, and overall performance, paving the way for future improvements with advanced closed-loop control techniques.

Code 3:

Closed Loop Prototype/ESC$_{C}losed_{l}oop.ino$

Results: Setup the Voltage Divider Hardware circuit for feedback. Configured the Analog Comparator to detect Zero Crossing timings and commutate the motor accordingly. Observed the trapezoidal waveform of the motor commutation sequence. At higher speeds the motor showed a tendency of stalling periodically due to the harmonic noise generated by the coils at high speeds inducing a voltage drop in the supply voltage.

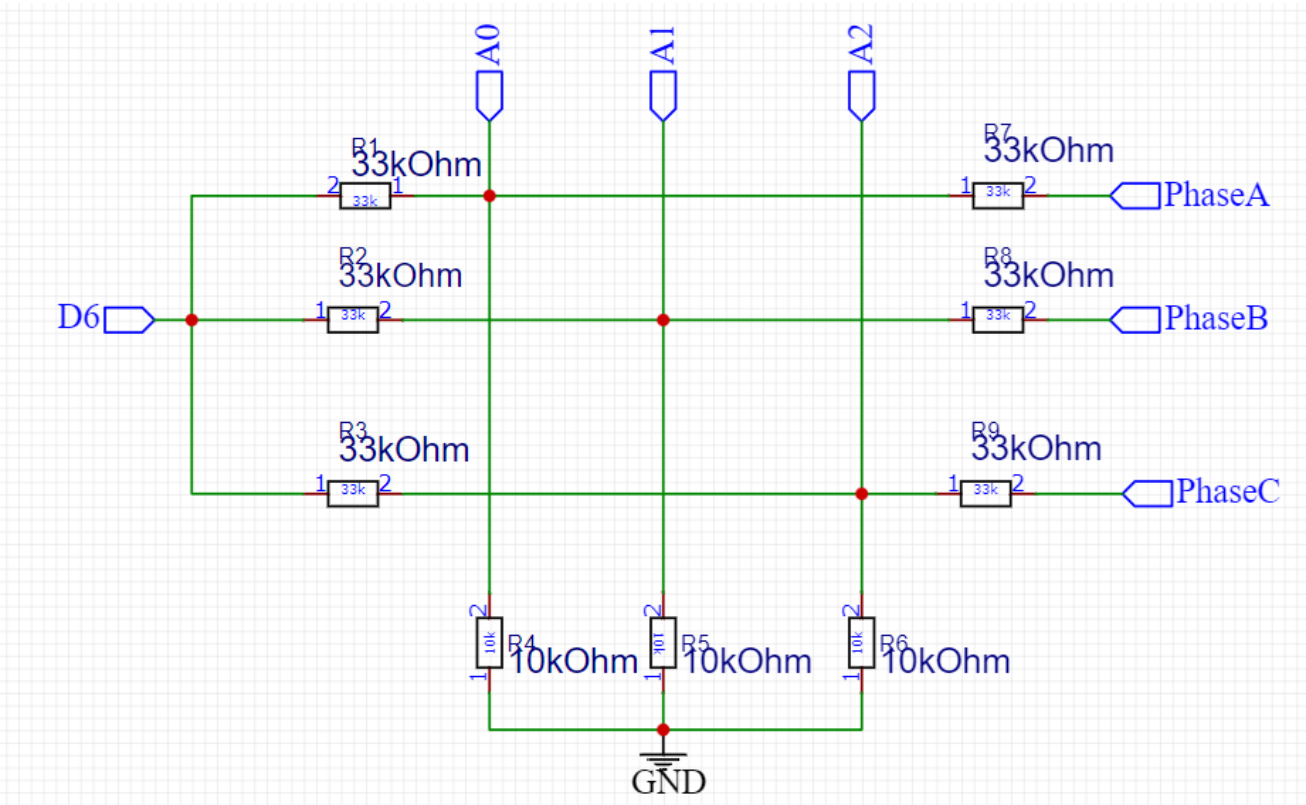

Figure 5: Back emf Voltage divider circuit

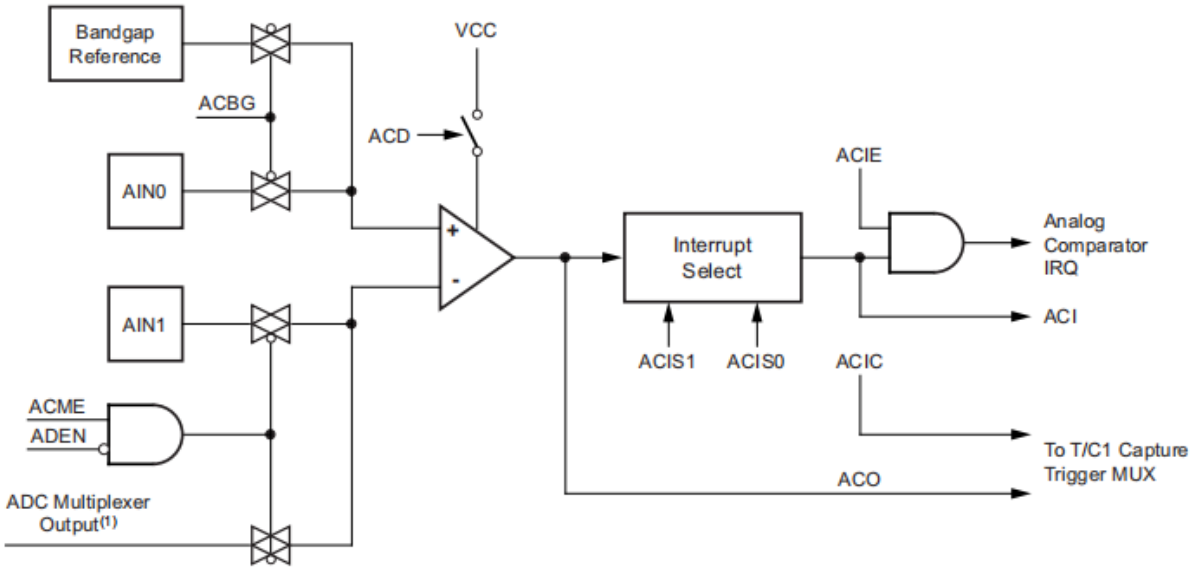

Figure 6: Analog Comparator onboard Atmega 328P

1.4 Adaptive Filtering for Back EMF Noise Reduction

The adaptive filter operates by dynamically adjusting both the confirmation threshold for valid zero-crossings and the minimum time interval between commutations based on the motor's speed and recent performance.

- **Dynamic Confirmation Threshold:** The system requires a series of consecutive valid signal readings before it confirms a zero-crossing. This threshold is adjusted according to the motor speed: at higher speeds, the filter requires fewer consecutive confirmations because the signal is more robust, while at lower speeds, more confirmations are needed to ensure the reliability of the signal.
- **Minimum Interval Between Commutations:** To avoid false triggers due to noise or rapid fluctuations, the system enforces a minimum delay between commutation events. This delay is adaptively derived from the average time interval between recent commutations, ensuring that the commutation frequency remains consistent with the motor's operational speed. However, this interval is bounded within a predefined safe range to prevent commutations from occurring too quickly or too slowly.
- **Feedback from Historical Timing Data:** The Analog Comparator ISR maintains a record of the most recent commutation intervals in a circular buffer. By continuously updating this history and computing a moving average, the system can smooth out the effects of transient fluctuations. This average then informs the calculation of the minimum allowed interval, enabling the adaptive filter to respond to changes in motor speed and performance in real time.
- **Noise Rejection:** If an unexpected signal is detected at any point, the system resets the confirmation counter. This prevents sporadic noise from being interpreted as a valid zero-crossing, thus maintaining reliable operation.

Code 4:

Closed Loop Prototype/ESC$_{C}losed_{l}oop_{A}daptive_{F}iltering.ino$

Results: Implemented an Adaptive filter by reducing the number of dynamic confirmations of zero crossing at higher speeds and increasing the number of iterations at lower speeds.Setup an adaptive delay by setting up a minimum interval between commutation steps to prevent rapid fluctuations due to the noise generated. These filtering mechanisms resulted in reduced stalling and slipping between

the commutation steps resulting in a lesser voltage drop of the supply voltage.

The Limited Timer Resolution of the 8 bit timer in the Atmega 328P microcontroller and the ISR latency causes slight voltage drop at higher speeds. These problems can be possibly eradicated by replacing the existing microcontroller with more efficient MCUs like STM32 which is more compatible for high speed Motor control applications requiring higher resolution and processing speeds.

Figure 7: Trapezoidal Waveform of 6 step Commutation of the motor