






Man-Machine-Interface Software Design of a Cotton Harvester Yield Monitor Calibration System

Mathew G. Pelletier ^{*}, John D. Wanjura  and Greg A. Holt 

Agricultural Research Services, United States Department of Agriculture, Lubbock, TX 79403, USA;

John.Wanjura@usda.gov (J.D.W.); Greg.Holt@usda.gov (G.A.H.)

* Correspondence: mathew.pelletier@usda.gov; Tel.: +1-806-746-5353

Received: 6 September 2019; Accepted: 9 October 2019; Published: 21 October 2019



Abstract: Several yield monitors are available for use on cotton harvesters, but none are able to maintain yield measurement accuracy across cultivars and field conditions that vary spatially and/or temporally. Thus, the utility of yield monitors as tools for on-farm research is limited unless steps are taken to calibrate the systems as cultivars and conditions change. This technical note details the man-machine-interface software system design portion of a harvester-based yield monitor calibration system for basket-type cotton strippers. The system was based upon the use of pressure sensors to measure the weight of the basket by monitoring the static pressure in the hydraulic lift cylinder circuit. To ensure accurate weighing, the system automatically lifted the basket to a target lift height, allowed basket time to settle, then weighed the contents of the basket. The software running the system was split into two parts that were run on an embedded low-level micro-controller, and a mobile computer located in the harvester cab. The system was field tested under commercial conditions and found to measure basket load weights within 2.5% of the reference scale. As such, the system was proven to be capable of providing an on-board auto-correction to a yield monitor for use in multi-variety field trials.

Keywords: yield monitor; weighing system; cotton harvesting

1. Introduction and Overview of the Research and Results of System Performance That This Technical Note Is in Support of

Several yield monitors are available for use on cotton harvesters, but none are able to maintain yield measurement accuracy across cultivars and field conditions that vary spatially and/or temporally. Thus, the utility of yield monitors as tools for on-farm research is limited unless steps are taken to calibrate the systems as cultivars and conditions change. This technical note details the man-machine interface software design for a harvester-based yield monitor calibration system for basket-type cotton strippers and is one of three technical notes that are in support of a master research paper covering the development and design of the calibration system. This technical note is presented, along with requisite supporting software source-code files, for the purpose of transferring the technology to the research community and general public. In addition to the source-code files, the technical notes provide documentation describing key strategies and methodology utilized in the design as well as background summary on the research.

In the process of evaluating the effects of production inputs and practices on crop yield and farm revenue, large scale field experiments are needed to investigate treatment effects across varying field conditions. Normally, these investigations require the use of additional labor and expensive ancillary equipment to weigh the crop harvested from a given area. The time required to collect this data reduces harvest productivity and efficiency. In order to reduce the dependence on ancillary equipment and labor, and increase the efficiency of data collection, yield monitors used on grain combines have

proven to be reliable tools in on-farm research efforts due to their consistent accuracy and lack of need for variety specific calibration. However, yield monitors used on cotton harvesters do not exhibit the same utility for on-farm research as they require frequent calibration when varieties or crop conditions change (Rains et al., 2002; Wilkerson et al., 2002; Robertson et al., 2006; Stewart et al., 2008; Taylor et al., 2014; Wanjura et al., 2014; Vories et al., 2019) [1–7].

Cotton yield monitors sense the flow of seed cotton inside conveying ducts or as the material passes into the basket or accumulator on the harvester. Yield flow sensors are generally of two designs: (1) Light attenuation (Givili, 1998; Wilkerson et al., 2001; Thomasson and Sui, 2000) [8–10] or (2) microwave reflectance (John Deere, 2010) [11]. In either case, the flow of material is related to the amount of light attenuated or microwave energy reflected by the flowing material. Material properties such as boll size, seed size, foreign matter content, lint turnout, fiber quality, and seed cotton moisture content are related to the accuracy of cotton yield monitors; all of which, except seed cotton moisture content, are cultivar specific properties (Head et al., 2009; Wilkerson et al., 2002; Wanjura et al., 2014; Vories et al., 2019) [2,6,7,12]. Setup and operation factors such as sensor alignment, stray light, temperature, and dust/foreign matter accumulation also affect cotton yield monitor performance (Wolak et al., 1999; Sassenrath-Cole et al., 1999, Sui and Thomasson, 2002) [13–15]. While uncalibrated systems can realistically reflect in-field variability (Thomasson and Sui, 2003) [16], calibration of cotton yield monitors is of utmost importance in producing accurate yield data.

Wanjura et al. (2015 and 2016) [17,18] describe the development and testing of a system for use on cotton harvesters that measures accumulated cotton weight inside a harvester basket; thus, facilitating the frequent calibration of cotton yield monitors without the need for expensive, time consuming, and often unavailable ancillary mobile scale equipment. The design of the system described by Wanjura et al. (2015 and 2016) [17,18] is briefly detailed herein with the main focus of this report being relegated to the final version of the electronic design of the system.

The harvester-based yield monitor calibration system (Wanjura et al., 2015 and 2016) [17,18] measures accumulated cotton weight in a harvester basket based on measurements of hydraulic pressure in the basket lift cylinder circuit. The system was designed and implemented on a John Deere 7460 (John Deere, Moline, IL) basket-type cotton stripper. A model relating hydraulic pressure in the lift cylinder circuit at a single basket position in the dump cycle was developed for basket weights ranging from 27 to 1633 kg (60 to 3600 lb.). The linear calibration model ($R^2 = 0.998$) was developed from 161 basket loads and exhibited RMSE of 9.9 kg (21.8 lb.) with mean absolute error of 0.44% (span). Hydraulic pressure was measured using a pressure transducer with 0–17,237 kPa (0–2500 psi) pressure range from Omega Engineering (PX409-2.5KG5V-EH, error specification $\pm 0.05\%$ FS = ± 8.62 kPa). A pulse-width modulated, proportional directional control valve (model SP08-47C, HydraForce Inc., Lincolnshire, IL) and solenoid operated check valves (model SV10-29, HydraForce Inc., Lincolnshire, IL, USA) were used in parallel with the harvester hydraulic system to raise the basket to the desired measurement position (13.7° from fully down) and hold it there once motion stopped. Two limit switches mounted at the rear of the harvester basket slowed and stopped the motion of the basket as it approached the target position. During the development of the linear calibration model, a magnetostrictive linear position sensor (MHC1400MN10E3V11, MTS Sensors, Cary, NC; error specification $\pm 0.04\%$ FS = 0.56 mm, repeatability $\pm 0.005\%$ FS = 0.07 mm) measured the extension of the lift cylinders to confirm that the limit switches could repeatably stop the basket at the desired position. Average lift cylinder extension during calibration model development was 160.53 mm (6.32 in) with standard deviation of 0.898 mm (0.035 in) which corresponded to a basket rotation angle of 13.7° $\pm 0.2^\circ$ from the down position.

Control of the calibration system hydraulic valves and data acquisition were accomplished with custom designed electronics [19], that was driven by custom embedded micro-controller software, [20]. A low-level microcontroller, and supporting circuitry [19], was mounted to a specially designed printed circuit board [19] that controlled the hydraulic valves during basket positioning and recorded pressure data once the basket was properly positioned. The hydraulic control algorithm raised

the basket into position through a twostep process whereby the pulse-width-modulated, PWM, directional-control-valve, DCV, raised the basket at 100% duty cycle until the basket passed the first limit switch. The PWM DCV duty cycle was lowered slowing the basket to about 25% of the original lifting speed as it approached the target position. The DCV was closed and the basket stopped when the basket passed the second limit switch. Simultaneously, the control algorithm closed the check valves on the lift cylinder circuit to isolate the static pressure in the lift cylinders and recorded the pressure after a brief stabilization period (2 sec). A 22-bit analog to digital converter was used to capture the pressure transducer analog-signal, with help from custom designed circuitry to minimize the influence of electrical noise and improve accuracy, [19]. A serial communication protocol was implemented between the embedded micro-controller software, [20], and the human-machine- interface, HMI, software, described herein, running on a mobile computer in the harvester cab. The HMI software, described herein, was programmed to calculate and display the weight of cotton in the basket using the pressure measured in the lift cylinder circuit and the initial calibration model. GPS position data was collected from a Greenstar 3000 receiver (John Deere, Moline, IL, USA) on the harvester and processed by the HMI software that used the GPS position data along with the user input of harvester width, to calculate the area from which the cotton in the basket was harvested. Seed cotton yield was calculated using the measured cotton weight and harvested area. Cotton ownership information (client, farm, and field) and machine header width (number and spacing of harvesting units) were recorded by the HMI software for each load measured.

The harvester-based yield monitor calibration system was field tested under commercial conditions on four producer-owned and operated cotton strippers. Weight measurement accuracy observed for the system during field testing was characterized by RMSE between 1.7 and 2.3% of span. Thus, it was concluded that the on-harvester calibration system would provide producers and researchers an accurate tool for use in (1) conducting on-farm research in which total plot yield is the evaluation metric, and (2) in calibrating cotton yield monitors without the need for costly and labor-intensive mobile scale systems.

The objective of this article is to describe the design of the man-machine-interface software that provides the main interface for the operator of the harvester to allow for interaction with the embedded micro-controller software [20] that provides the control and interfacing to the electronic data-acquisition and control sub-system [19,21] that all together in combination forms the cotton harvester yield monitor calibration system.

2. Man-Machine-Interface Software Design

2.1. Background Development Tools

The software discussed in this technical note provides the bridge between a hardware micro-controller (machine controller) and the human operator. Hence, the use of the HMI label, which we've borrowed from common usage from the Industrial Control sector given they are basically doing the same thing (human operator interface that talks to a micro-controller (PLC) that actually provides the control of the machines. In the system discussed herein; the man-machine-interface, MMI, software design was designed around an event driven windows style program using the cross-platform C++ programming libraries and integrated-development-environment, IDE, provided by QT Company, hereafter known as QT [22]. One of the main advantages of using QT is that it allows for a write-once run anywhere model that is supported across a great number of operating systems, OS, and hardware platforms such as (Linux ARM, Linux x86, Machintosh OS, IOS, Windows, Android). The QT libraries have built into them a very clean event driven model that is based around a signal-slot paradigm whereby when a device or function wants to generate an interrupt-service request it issues a signal to the library that then calls any associated slot-flagged functions. Utilizing this technique, the programmer can link a function, designated as a slot, that will be called every time the signal is emitted by any other function or library request for servicing. This methodology provides a clean

event driven system that can easily be leveraged to build a windows style program that is completely event driven such that any user input, via mouse or keyboard, will be rapidly responded to by the QT library and slot-associated functions provided by the programmer. In this manner, QT provides the underlying architecture for a generic idle, till event occurs, scan loop. Using the QT paradigm, the yield calibration system man-machine-interface software was constructed so that all the main functions sit idle till an event is triggered that fires that corresponding function. Some of the key events the software utilizes that trigger the slot-denoted functions are:

- Serial-port communication reception of bytes
- Asynchronous functions responding to events
- User clicking or pressing on form buttons or typing in text boxes
- QT Timers

Another key feature that enables for rapid program development using QT is its IDE, which features an easy to use form layout tool that automatically links slot-functions to appropriate buttons, text boxes, radio-buttons, and other assorted graphical-user-interface, GUI, form elements. As the target of the system was to run on a harvester, the OS of choice was to utilize Linux as it has the best support for embedded hardware that is most appropriate for non-desktop industrial environments.

2.2. User Interface

Utilizing the QT platform, a series of forms were developed that shaped the operator interface for the software. The main screen is shown in Figure 1. The main setup screen is shown in Figure 2.

1	2	3	4
1 Farm	Field	Distance...	Area (acr...
2			

1	2	3	4
1 Load #	Field ID	Yield (lbs...	Weight (l...
2			
3			
4			
5			
6			
7			
8			
9			
10			

1	2	3	4
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			

Figure 1. The main screen that provides the main operator interface screen provides the main user interface to control the harvester yield monitor calibration system.

Figure 2. The setup screen that allows for user input so the software can associate the client-farm-field information with the subsequent data-acquisitions.

2.3. Software Design

The software was designed so that upon program startup (boot), the initialization routines are automatically run, by utilizing default and previously saved settings. This boot-up was designed as a two-step process, as shown in Figure 3. The main core of the initialization routines are:

- Instantiate all supporting utility classes,
- Configure QT event driven signal-slot connections for each of the serial ports for (global satellite positioning system (GPS), Data-Acquisition),
- Configure and verify serial communication to:
 - GPS
 - Data-Acquisition
- Load default standard saving locations, filenames, and data-base settings,
- Load farm-field setup screen and get user input as to specifics by which to tag all collected data,
- Delay using QT timer long enough to allow for GPS to establish connection and verify connection,
- Verify connection to the micro-controller that provides the data-acquisition and weigh controller functionality.

From the system's main state-diagram, Figure 3, the process starts out with a couple of initialization steps and then after it loads default parameters from a file and verifies that GPS and the data-acquisition system are available on the serial ports, it then proceeds to the Idle state block. The Idle state-block provides the main loop for the system design and is provided by the "Main_Window" QT event loop, Figure 4. In Idle mode; the system basically waits until an event happens and then reacts to this event. For example; the "GPS_Comm" state block, Figure 4, shows the state-transition from "GPS_Comm" reception of serial data to appropriate "Main_Window" functions that are fired upon the generation of a signal occurring on receipt of a message from "GPS_Comm" that includes the serial bytes received. These bytes are then parsed to form a complete GPS message that is encapsulated in the "GPS_Data" passed to a processing function, "Receive_GPS", in the class "cGPS_utils". The expansion of "GPS_Comm" block, of Figure 3, is detailed with a state-diagram in Figure 4. "GPS_Comm" starts out in the "Main_Window" event loop where it waits on reception of serial bytes from the GPS, where upon reception it automatically fires the serial-data received signal which in turn fires the event driven slot-linked function "Read_data_GPS". Upon activation with the serial data, "Read_data_GPS" parses the data and ensures it contains a complete GPS National Marine Electronics Association, (NEMA) 0183 RMC message [23] that it then passes in a function call to the "Receive_GPS" function, that is

located in the C++ class “cGPS_utils” where the location, velocity, and time information is extracted and saved into relevant data-structures for later processing.

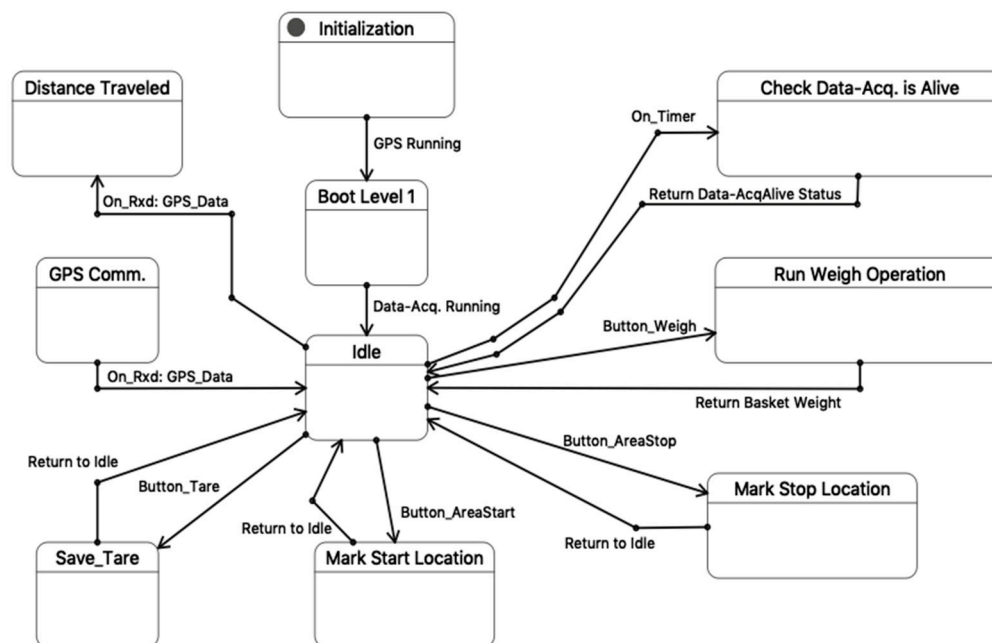


Figure 3. The state diagram that describes the flow for the main program loop in the man-machine-interface software.

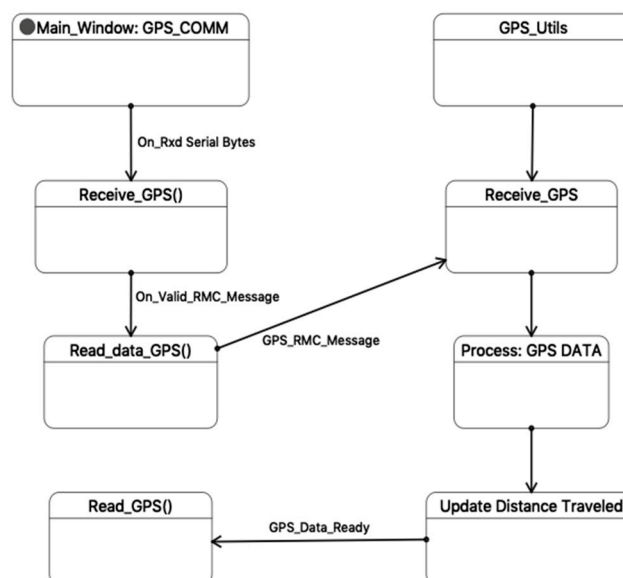


Figure 4. The state-diagram chart for the global-positioning-satellite-system, GPS, serial receive and processing functions that perform key functions inside the main processing loop of the man-machine-interface software.

The “Receive_GPS” function, Figure 4, removes the waiting serial bytes from the GPS serial-port buffer and applies it to the end of its GPS data-buffer; then parses the string looking for the first occurrence of “\$RMC”, which is the start of the NEMA-0183 RMC data-message [23] and checks if there is a carriage-return following \$RMC that would indicate a complete message was obtained. This process is detailed in the flow-chart shown in Figure 5. Once a complete RMC message is found, the function then calls into the “cGPS_utils” class function “Receive_GPS”, Figure 4. Which parses

the RMC message for position (latitude, longitude), Coordinated-Universal-Time, UTC-time, travel velocity estimation and direction of travel [23]. This data is saved into short-duration history, for distance tracking purposes and later data-set creation and saving into the data-base.

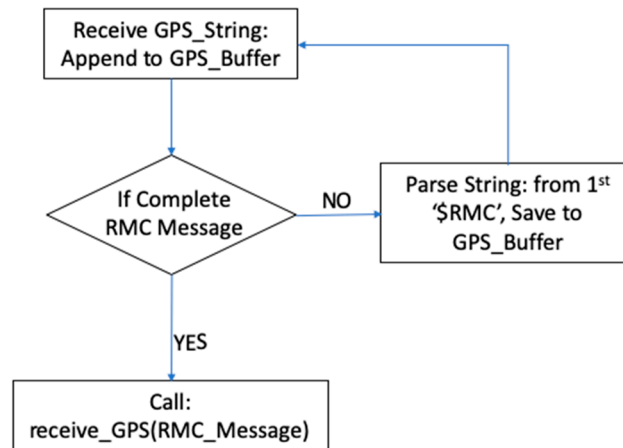


Figure 5. The flow chart for the GPS serial receive function that operates inside the main-loop of the man-machine-interface software.

2.4. GPS Distance Computation

With the valid RMC GPS location position fix; the “Receive_GPS” function passes data to the “GPS_DATA” function for further processing, which uses the most recent current position fix along with its saved position history to compute distance traveled using the Haversine formula, (Goodwin, 1910) [24]. Hence at every valid GPS RMC message, the distance from the original starting point, and the last point is computed. This was done to provide a real-time display estimate of the area covered during the harvest operation between operator flagged points of interest noted by operator pressing “Area Start” and “Area Stop” buttons on the user interface, Figure 1. The Haversine formula was used to compute the shortest distance following the arc of the earth’s surface, Figure 6.

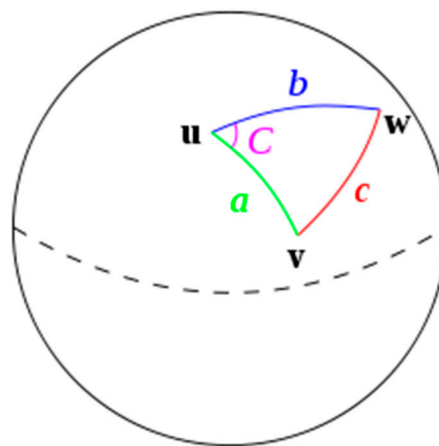


Figure 6. Spherical triangle solved by the law of haversines, equations [1–3].

The haversine formula is well conditioned even for points as close to each other as a few meters. The formula is designed to calculate the great-circle distance between two points and is detailed in Equations (1)–(3). In the software, the haversine formula is computed in the source code located in the “cHaversine” class in the function “gps_distance”.

$$a = \sin^2\left(\frac{\Delta\varphi}{2}\right) + \cos(\varphi_1) \cos(\varphi_2) \sin^2\left(\frac{\Delta\lambda}{2}\right) \quad (1)$$

$$c = 2 \operatorname{atan2}\left(\sqrt{a}, \sqrt{1-a}\right) \quad (2)$$

$$d = R * c \quad (3)$$

where

φ_1 is latitude of first location

φ_2 is latitude of second location

λ_1 is longitude of first location

λ_2 is longitude of second location

$\Delta\varphi$ is latitude difference between points 1, 2

$\Delta\lambda$ is longitude difference between points 1, 2

R is earth's radius (mean radius = 6371 km)

$\operatorname{atan2}()$ is four quadrant function that computes $\tan^{-1}()$

c is angular distance in radians

a is the square of half the chord length between the points

d is the distance between the two GPS points (m)

2.5. GPS Distance Integration

After completion of GPS distance traveled computation, the distance traveled is then relayed back to the “main-window” “Idle” loop, Figure 4, that updates the display with the total area covered by the harvester since the last time the “Area Start” button was pressed, Figure 1. How the main “Idle” event loop is processed is detailed in Figure 7, where it is shown that upon the press of the “Area Start” button, (shown on state diagram as “Button: Mark_Start_Location”) that a call is made into “cGPS_utils” class function “mark_start_GPS()”, which saves the current GPS location as the starting location point by which to measure area harvested. It also writes a flag into the stored dataset for post-processing notification as to where the start of the harvested plot was located.

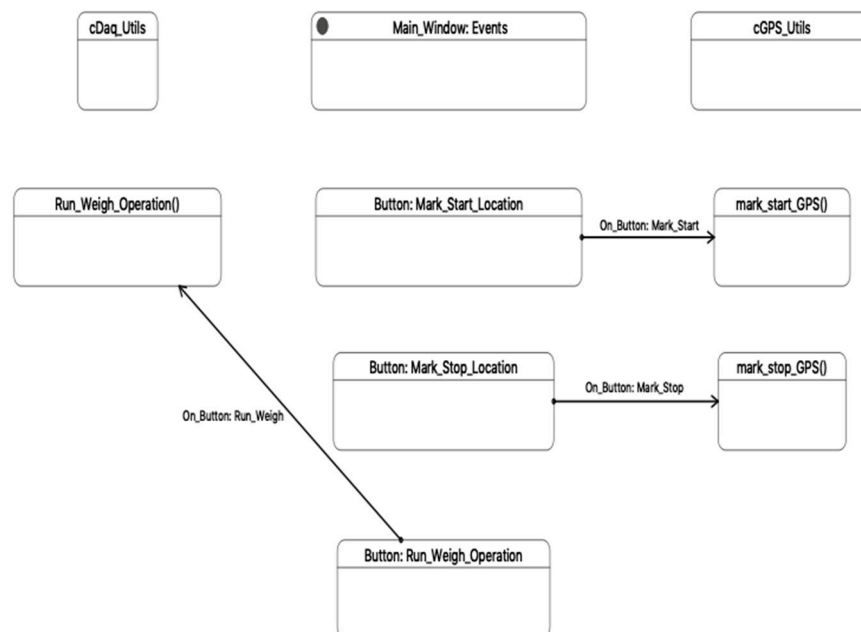


Figure 7. The state-diagram chart for the GPS serial receive function that forms one of the key events the main loop for the man-machine-interface software.

2.6. GPS Area Marking

Similarly, the “Area Stop” button fires the “mark_stop_GPS” function which marks the final position location and terminates the distance logging feature. The “Weigh Basket” button, Figure 1, triggers the state “Run_Weigh_Operation”, which is a multi-step process that starts by automatically calling “Area Stop” event that then proceeds by sending a serial request string to the micro-controller to perform a weigh cycle. Upon sending the serial request to the micro-controller to perform a weigh-cycle, the “Run_Weigh_Operation” function then exits into the main “Idle” loop to await an asynchronous response from the micro-controller. Upon completion of the weigh operation, the micro-controller responds back to the HMI software with serial data form the weigh operation. The reception of the serial data triggers a serial-data-received event that contains the basket weigh information, there-by completing the full weigh cycle. As the weigh cycle takes some time to complete, the program reverts back to “Idle” mode while it awaits the weigh data. The software design to accommodate this multi-system distributed asynchronous event driven design is accomplished using the same topology that was used to interface to the GPS system, Figure 7. In the case of the micro-controller interface; the transition occurs upon receipt of serial data from the micro-controller which causes a serial event trigger “readData_daq” whereby the serial handling function for the micro-controller checks for valid message and when found relays the cleaned up message to “readDAQ” function, which then posts the weigh information onto the main screen and saves the data to data-base along with the previously saved “Area Stop” GPS location. The final distance is then computed using the Haversine formula on the GPS location points marked at time the button events triggered by the “Area Start” and “Area Stop” button press occurrences. This GPS measured harvested travel distance is then multiplied by the user provided harvested width where the product provides the harvested area. The harvester’s width is set in the initial setup screen, Figure 2, which is used at the start of the harvesting session by the user to provide key information, such as the farm, client, and field names and any notes of relevance the user wants to save along with the data-set. An additional parameter of interest on the setup screen in Figure 2 is the wind conditions, (calm, gusty), the user is currently having to contend with. The wind-condition parameter provides both a useful note into the data-set, as the harvester’s basket presents a sizable area that is impacted by a significant wind-load impacting the accuracy of the weighing operation; as well as allows the software to request that the micro-controller software [20] utilize a longer settling time to help ensure an accurate weigh operation is conducted.

2.7. Software Design Summary

This review covered the main aspects of the software source code; for specific details on the full code implementation, a list of all the functions that make up the man-machine-interface software are shown in Figure 8 and a full copy of the documented source code is attached in supplemental files.

While there is a great deal more code in this software package, the rest of the code is self-explanatory and well documented in the attached files and is left to the reader to peruse.

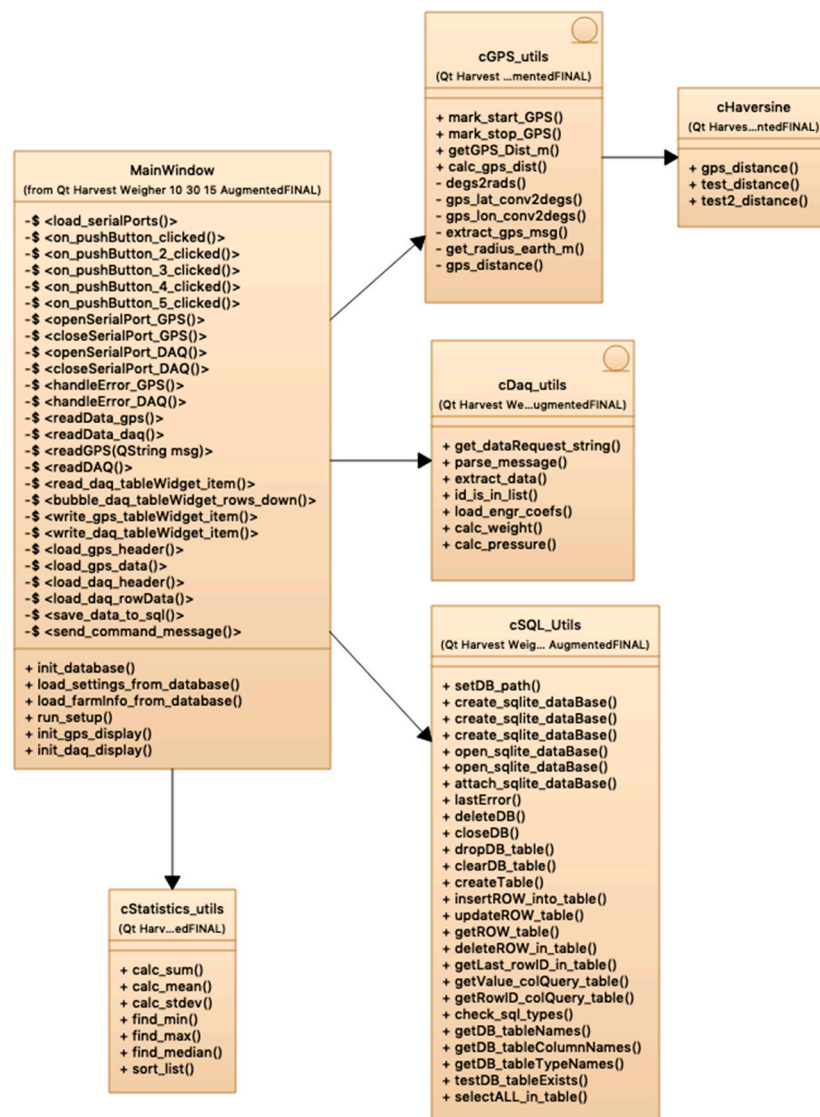


Figure 8. The C++ class definitions used in the man-machine-interface software. In the figure's class specifications, the leading '+' signifies public functions. While the leading '-' signifies private functions and the leading '\$' signifies event response function, SLOT, that are called and used by the windowing event loop structures. The SLOT functions provide the equivalent of an interrupt service routine that frees up active process from having to continuously poll various resources for data event occurrences. The class structure is flat with no inheritance by any of the classes; the arrows between classes only indicate which class is calling other classes by means of publicly visible functions.

3. Conclusions

The man-machine-interface software design covered in this technical note was tested in conjunction with the electronic design [19], the vehicle transient surge protection circuitry [21] and the embedded micro-controller firmware software [20]. The system was proven both on the laboratory bench and on the harvester under field conditions, and was found to be stable, robust, accurate, and suitable for use in on-farm research efforts [17,18]. With some minor software modifications, the system could be easily extended to dynamically correct yield monitor data as well. Future notes are planned to cover both the hydraulic circuit design as well as the companion software implementation that provides the industrial PC operator interface code.

Supplementary Materials: The following are available online at <http://www.mdpi.com/2624-7402/1/4/37/s1>. The software source code files are available along with this technical note online and is released into the public-domain as Open-Source Software under MIT Open-source license. The code is written for QT windowing API and was compiled using QT Creator IDE using QT-5 Application-Programming-Interface, API.

Author Contributions: Conceptualization, M.G.P. and J.D.W.; Methodology, M.G.P. and J.D.W.; Software, M.G.P.; Validation, J.D.W. and M.G.P.; Formal analysis, M.G.P. and J.D.W.; Investigation, M.G.P. and J.D.W.; Project administration, G.A.H.

Funding: This research received no external funding.

Conflicts of Interest: Mention of a product or trade-name in this article does not constitute an endorsement by the USDA-ARS over other compatible products. Products or trade names are listed for reference only. USDA is an equal opportunity provider and employer.

References

1. Rains, G.C.; Perry, C.D.; Vellidis, G.; Thomas, D.L.; Wells, N.; Kvien, C.K.; Dales, H.D. *Cotton Yield Monitor Performance in Changing Varieties*; ASAE Paper No. 021160; ASAE: St. Joseph, MI, USA, 2002.
2. Wilkerson, J.B.; Moody, F.H.; Hart, W.E. Implementation and field evaluation of a cotton yield monitor. *Appl. Eng. Agric.* **2002**, *18*, 153–159. [[CrossRef](#)]
3. Robertson, B.; Cordell, M.; Matthews, S.; Groves, F. Utility of Yield Monitors for On-farm Research. In *Proceeding 2006 Beltwide Cotton Conferences*; National Cotton Council: Memphis, TN, USA, 2006; pp. 1756–1758.
4. Stewart, A.M.; Wright, I.R.; Deville, S.F.; Woolam, B.W. Comparison of On-Farm Cotton Variety Trial Results when Using Yield Monitors vs. Weigh Wagons. In *Proceeding 2008 Beltwide Cotton Conferences*; National Cotton Council: Memphis, TN, USA, 2008; pp. 69–71.
5. Taylor, R.; Porter, W.; Boman, R.; Osborne, S.; Henderson, W.; Buschermohle, M.; Barnes, E. Using yield monitors to evaluate cotton variety tests. In *Proceeding 2014 Beltwide Cotton Conferences*; National Cotton Council: Memphis, TN, USA, 2014; pp. 494–498.
6. Wanjura, J.D.; Kelley, M.S.; Taylor, R.K.; Porter, W.M.; Barnes, E.M.; Pelletier, M.G.; Holt, G.A. Evaluation of a cotton stripper yield monitor. In *Proceeding 2014 Beltwide Cotton Conferences*; National Cotton Council: Memphis, TN, USA, 2014; pp. 481–493.
7. Vories, E.D.; Jones, A.S.; Meeks, C.D.; Stevens, W.E. Variety Effects on Cotton Yield Monitor Calibration. *Appl. Eng. Agric.* **2019**, *35*, 345–354. [[CrossRef](#)]
8. Gvili, M. Cotton yield sensor produces yield maps. In *Proceeding 1998 Beltwide Cotton Conferences*; National Cotton Council: Memphis, TN, USA, 1998; pp. 1655–1657.
9. Wilkerson, J.B.; Moody, F.H.; Hart, W.E.; Funk, P.A. Design and Evaluation of A Cotton Flow Rate Sensor. *Trans. ASAE* **2001**, *44*, 1415–1420. [[CrossRef](#)]
10. Thomasson, J.A.; Sui, R. Advanced optical cotton yield monitor. In *Proceeding 2000 Beltwide Cotton Conferences*; National Cotton Council: Memphis, TN, USA, 2000; pp. 408–410.
11. Deere & Company. *Cotton Mass-Flow Sensor*; Deere & Company: Moline, IL, USA, 2010. Available online: http://salesmanual.deere.com/sales/salesmanual/en_NA/cotton_harvesters/2011/feature/pickers/ams/cotton_mass_flow_sensor.html (accessed on 8 August 2019).
12. Head, J.C.; Wilkerson, J.B.; Hart, W.E.; Allen, P.B. Identification and quantification of cotton yield monitor errors. In *Proceeding 2009 Beltwide Cotton Conferences*; National Cotton Council: Memphis, TN, USA, 2009; pp. 374–377.
13. Walak, F.J.; Khalilian, A.; Dodd, R.B.; Han, Y.J.; Keshlkin, M.; Lippert, R.M.; Hair, W. Cotton yield monitor evaluation, South Carolina—Year 2. In *Proceeding 1999 Beltwide Cotton Conferences*; National Cotton Council: Memphis, TN, USA, 1999; pp. 361–364.
14. Sassenrath-Cole, G.F.; Thomson, S.J.; Williford, J.R.; Hood, K.B.; Thomasson, J.A.; Williams, J.; Woodard, D. Field testing of cotton yield monitors. In *Proceeding 1999 Beltwide Cotton Conferences*; National Cotton Council: Memphis, TN, USA, 1999; pp. 364–366.
15. Sui, R.; Thomasson, J.A. Test of temperature and stray-light effects on mass-flow sensor for cotton yield monitor. *Appl. Eng. Agric.* **2002**, *18*, 429–434.
16. Thomasson, J.A.; Sui, R. Mississippi cotton yield monitor: Three years of field-test results. *Appl. Eng. Agric.* **2003**, *19*, 631–636. [[CrossRef](#)]

17. Wanjura, J.D.; Pelletier, M.G.; Holt, G.A.; Kelley, M.S. A harvester-based calibration system for cotton yield monitors. In *Proceeding 2015 Beltwide Cotton Conferences*; National Cotton Council: Memphis, TN, USA, 2015; pp. 635–647.
18. Wanjura, J.; Pelletier, M.; Holt, G.; Kelley, M. Reliability testing of an on-harvester cotton weight measurement system. In *Proceeding 2016 Beltwide Cotton Conferences*; National Cotton Council: Memphis, TN, USA, 2016; pp. 658–670.
19. Pelletier, M.G.; Wanjura, J.D.; Holt, G.A. Electronic Design of a Cotton Harvester Yield Monitor Calibration System. *AgriEngineering* **2019**, forthcoming.
20. Pelletier, M.G.; Wanjura, J.D.; Holt, G.A. Embedded Micro-controller Software Design of a Cotton Harvester Yield Monitor Calibration System. *AgriEngineering* **2019**, *1*, 485–495. [[CrossRef](#)]
21. Pelletier, M.G.; Wanjura, J.D.; Holt, G.A.; Funk, P.A. Methods for Protecting a Personal Computer and Data Acquisition Electronics Installed on Mobile Equipment. *AgriEngineering* **2019**, *1*, 4–16. [[CrossRef](#)]
22. QT Company. *QT Version 5.5*; QT Group Plc: Helsinki, Finland, 2014. Available online: <https://www.qt.io> (accessed on 6 August 2019).
23. National Marine Electronics Association. *NEMA 0183 V4.0 Stand*; National Marine Electronics Association: Severna Park, MD, USA, 2008.
24. Goodwin, H.G. The haversine in nautical astronomy. *Nav. Inst. Proc.* **1910**, *36*, 735–746.



© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).