A Mobile Device to Quantify the Forces and Angles Required to Overcome Knee Arthrofibrosis in the Clinic

---

A Thesis

Presented to

the faculty of the School of Engineering and Applied Science

University of Virginia

In Partial Fulfillment

of the requirements for the Degree

Master of Science in Systems Engineering

Loi Huynh

January 20, 2016

CONTENTS

ABSTRACT

Knee arthrofibrosis, wherein a patient cannot fully flex the knee, is encountered after 4 - 35% of ligament surgeries. It is treated either by manually manipulating the knee to break scar tissue or scheduling further surgery. The work herein sought to design, build and evaluate a means of quantifying the forces and angles required to overcome arthrofibrosis during knee manipulation. Physicians are thought to rely upon several cues including the a) initial resting angle of the upper and lower leg, b) torques and angles at the break point of scar tissue, and c) maximum angle and torque following manipulation. To quantify these cues, the device was built to be mobile and attach to a common knee brace. Fixtures were designed to afford normal physician postures and manual control of torque. The device utilizes load cells, rotary potentiometers, 3D printed parts and integrated circuits. Graphical user interfaces on a mobile phone allow in-the-loop evaluation and a database is used for subsequent, off-line analysis. A preliminary clinical study with two participants shows the device can measure maximum torque of 31 Nm, rotational angles of 130 degrees, and combine both measures to differentiate knee stiffness over a range of motion.

1. INTRODUCTION

Arthrofibrosis is a common problem after knee ligament injury, traumatic injury, or elective surgery, such as total knee arthroplasty. Basically, the knee tightens so as to decrease the range of flexion. Several factors contribute to its development, including intraoperative technical errors, severity of injury, timing of surgery, delayed post-operative rehabilitation or prolonged post-operative immobilization, post-operative infection [1], or development of heterotopic ossification [2], [3]. Regardless of cause, the result is to significantly decrease a person's ability to ambulate and perform activities of daily living. For example, knee flexion of less than 90 degrees (measured as in Fig. 1 whereby the angle of lower leg is judged relative to a reference line extended from the upper leg, when upper leg is positioned perpendicular to the patient's position lying on a table) affects one's ability to sit, ascend, and descend [3] and knee flexion of a minimum of 67 degrees is required to maintain the swing phase of normal gait [4].

Treatment options for knee arthrofibrosis include manipulation under anesthesia (MUA), arthroscopic versus open lysis of adhesions, or revision total knee arthroplasty if the inciting event stems from joint replacement. Non-operative interventions are always the first-line of therapy; therefore, manipulation under anesthesia is attempted prior to surgical management. While the precise timing of post-operative manipulation under anesthesia (MUA) remains the subject of on-going study [5], [6], the basic procedure is to flex the knee 7-8 times to reach 120-130 degrees [7]. The desired outcome of MUA is to match an average gain in ROM of 30-50 degrees [5], [8] knowing that the failure rate to achieve this is 6.7% [9]. MUA is not without the risk of complications, which can be attributed to the procedure or to the post-operative period and include hemarthrosis, wound separation, supracondylar femur fracture, and post-operative pulmonary embolism [6], [10]. For these reasons, among others, a standardized and consistent means of conducting and monitoring the procedure are desired.

At present, the main method of quantitative assessment of the knee is to use a device such as a Medmetric KT-1000 arthrometer to assess cruciate ligaments for tibiofemoral translation in an anterior-posterior direction. While related to the topic at hand, these types of devices do not quantitatively

evaluate knee stiffness through a range of motion (ROM) of flexion and extension. Other devices do focus on measuring ROM. For example, the *Smart Knee* (Bend Labs, Inc.) can track knee angle in real-time using a bend sensor along with an integrated circuit that includes an accelerometer, gyroscope, and magnetometer [11]. The accelerometer can measure forces, but of foot impact. The device is used mainly for coaching regular exercise as well as rehabilitation. Furthermore, Dong et al. used a magneto-rheological damper to create variable resistance to supervise the exertion of force on joints of the elbow, hips, and knees [12]. They used a linear displacement sensor to measure range of motion, where the center of rotation of the sensor matched the center of rotation of interest. Further measures of displacement have employed ultrasonic transducers [13].

The work herein sought to build a system customized for the procedure of knee manipulation under anesthesia, with goals of maintaining low cost yet sufficient dynamic range, accuracy, and resolution. Since no such system exists, the first step was to understand the psychomotor cues upon which physicians rely to make clinical judgments. Then, a mobile system, attached to a common knee brace, was built to measure angles and torques about the knee, from which stiffness could also be derived. Fixtures were designed to afford normal physician hand positions and postures relative to the patient. Preliminary experiments were conducted with two participants undergoing MUA in the clinic, with the goal of determining if the prototype could meet the requirements.

2. METHODOLOGY

a. IDENTIFYING KNEE MANIPULATION PROCEDURES AND CUES

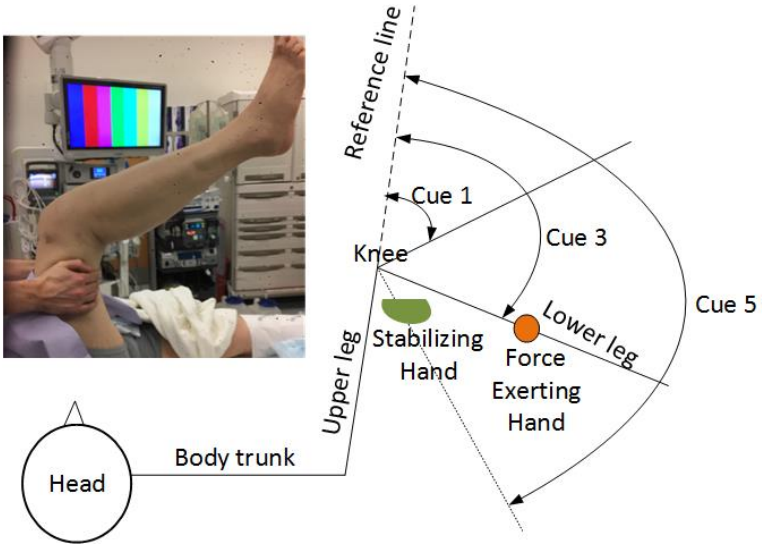

Fig 1. The upper left image shows a reference knee to undergo manipulation. The corresponding schematic shows cues to be measured and positions of the physician's force exerting and stabilizing hands.

To better understand the procedures and cues underlying knee manipulation, we conducted informal interviews and observations with physicians. Overall, the procedure involves holding a patient's upper leg at 90 degrees relative to the table upon where he is lying with a cupped arm, so that the lower leg is allowed to relax via gravity to its resting position, Fig. 1. The physician then starts to flex the lower leg until she or he perceives resistance, after which, with greater force she or he chooses whether to break through scar tissue. The torque exerted on the knee becomes larger until the scar tissue breaks. Following this initial break, the knee will begin to move freely until it reaches a larger angle of flexion. After that, the physician may continue to push the lower leg in exerting greater torque on the knee. The knee manipulation procedure concludes when the physician feels that the knee's flexion has improved and/or decides not to exert greater forces out of fear of causing damage. This marks the maximum angle of flexion. In the case where the physician is uncomfortable in breaking through scar tissue, the procedure concludes and other treatments become more promising.

Based on these observations, we isolate, describe and quantify three types of psychomotor cues below.

Cue 1. Relative angle of upper and lower leg at its initial resting position. When the physician moves the upper leg so that it is perpendicular to the table upon which the patient is lying, the lower leg will want to naturally flex due to gravity to a position. This angle, as shown in Fig. 1, is measured relative to the reference line, which is parallel to the upper leg. For a person with knee stiffness, this is typically about 0-20 degrees as opposed to 120-130 degrees in an otherwise healthy individual.

Cue 2. Torque and angle at the point where scar tissue is encountered, which blocks motion and may be broken by the physician. When the knee is moved from rest to the angle where scar tissue is encountered, the physician often faces resistance to move past the scar tissue break point. We quantify this resistance by considering torque (rotational moment), angle, and these two relative to each other. The rotational moment is the force imposed by the physician multiplied by the length of the lever arm where force is exerted. If too much force is required, physicians may stop the procedure.

Cue 3. Maximum torque and maximum angle. The maximum angle signifies the range of motion of the knee near the point it can no longer be moved in flexion. Cue 3 is defined similarly to Cue 2, except it is greater if the scar tissue has been broken and equivalent otherwise.

b. Functional Requirements and Design Tradeoffs

- Ease of donning: The device should be easily and rapidly put on and removed from the patient's leg to cope with the pace of the operating room.
- Firm attachment: The device must attach firmly to the leg, so that force and angle are consistent during the procedure.
- Mobility and weight: The device should be lightweight (less than 1 kg) and compact. All 3D printed material should be minimized to lower weight and cost.
- Feedback to physician operators: The device must provide the instantaneous force and angle as well as the maximums during the session and from the prior session.
- Learnability: The device should be easy for the physicians to learn to operate and not require a technical background.

- Untethered nature and battery life: The device should be untethered with a battery that lasts at least 1 hour to accommodate a single clinical procedure.
- Dynamic range versus resolution in force: The maximum force applied must be great enough to withstand that which physicians will exert on the brace, which is translated to the knee, yet differentiate small changes.
- Dynamic range versus resolution in angle: The device should be able to measure angles from 0 to 130 degrees, with a resolution of one degree.
- Low cost: The total cost of all components, excluding the commercial brace itself, should be less than $500.

c. APPARATUS

The knee measurement device was built to be mobile and measure rotational angles and torques exerted on the knee, from which stiffness could also be derived. The device includes sensors and electronics, which are attached to a common knee brace via 3D, printed superstructure. Fixtures were designed to afford normal physician hand positions and postures. In particular, as used in practice, the attending physician presses on the u-shape 3D printed piece, Fig. 2A. The force is read by two parallel beam load cells, positioned as cantilevers between the u-shaped fixture and the superstructure attached to the knee brace, while the angle of the superstructure pieces representing the lower and upper leg are simultaneously recorded via a rotary potentiometer. The electrical signals flow through amplifiers and into a microprocessor in a small box connected to the brace. From there, data are transmitted via Bluetooth wireless and displayed in real-time on a mobile phone as well as being logged to a database. The use of these the linear load cells (about $7 USD each) was done on the basis of cost, as rotary force transducers can exceed $2,000-5,000 USD.

Brace. The knee brace (Donjoy LEGEND-SE4, California USA) is composed of four straps, steel beams along both sides and a rotational pivot about their center. The brace can span -10 degrees extension to +130 degrees flexion.

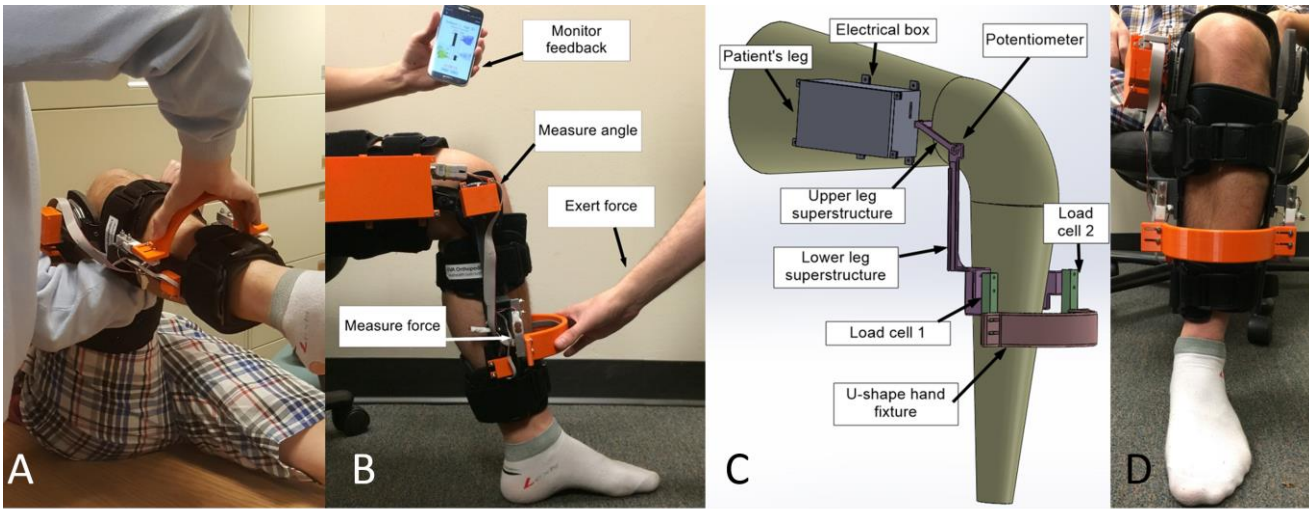

Fig 2. Design of the electronic brace. Panel A shows how the device is used in practice. In panel B, the device's components are given relative to the position of the physician's hand. Note that in actual operation of the device, the foot is in the orientation in the air as in Fig. 1. In panel C, the 3D printed components that constitute the superstructure for attachment to the knee brace as well as the electrical box and physician's hand fixture for exerting force. In panel D, a straight on version of the device is given.

3D Printed Parts. A superstructure was designed to attach to the knee brace so as to hold and conceal the sensors and electronics, and provide the hand fixture by which the physician exerts force. The geometry was built using computer aided design software (Solidworks, Version 2015, Dassault Systems, Massachusetts USA) and printed using a 3D printer (Stratasys, uPrint SE Plus, Minnesota USA). The superstructure parts were printed along with the electronics box, hand fixture, and two small boxes to protect the potentiometer and wires near the load cell, Fig. 2B. The electrical box has wall thickness of 2 mm, and width and height of 13 and 7.5 cm, respectively. Two superstructure attachments (upper and lower leg superstructures in Fig. 2C) connect with each other, and to the steel of the brace, via the potentiometer to measure their relative angle. The hand fixture's thickness is 0.8 cm to prevent it from bending. The elliptical shaped holes on either side of its base lets the brace fit different sized legs.

Sensors and Electronics. Components of this subsystem include a microprocessor, a Bluetooth communication module, a potentiometer, two load cells, two force amplifiers, and a rechargeable battery. The Arduino Uno Microcontroller (Adafruit R3, Dev-11021, Ivrea Italy) is a 16 MHz processor with 6 analog and 14 digital inputs, sufficient to connect with the sensors and Bluetooth module. The Bluetooth

wireless serial port module (Phantom Yoyo JY-MCU, China) transmits data from the microprocessor to the mobile phone. A voltage divider aligns the 3.3 volt signal of the Bluetooth JY-MCU with 5 volt operation of the Arduino. The Bluetooth module takes receiver and transmitter signals from two digital pins of the Arduino for broadcast to the mobile phone at 10 samples per second.

A rotational potentiometer (Honeywell 53C32K-ND, New Jersey USA) of 2 kilo-Ohms attached to the upper and lower leg superstructure pieces measure the rotation of the knee. Its voltage output spans a range of 318 degrees. In addition, two parallel beam load cells (HTC Sensor TAL220, Colorado USA) utilize strain gauges on bars to transduce forces up to 10 kg, or 98 N. An ultimate overload of 150% can be applied and the unit's error is 0.05%. Its form factor is 8 cm long by 1.2 cm both high and wide, and its internal circuit utilizes a Wheatstone bridge. Its electrical output of change in resistance is amplified and filtered through a 24-bit analog to digital converter (AVIA Semiconductor HX711, Colorado USA). The sampling rate was set at 10 samples per second.

Mobile phone and Battery. The mobile phone (Samsung Galaxy S4 with Android operating system, model GT-I9500, Seoul, Korea) with Bluetooth wireless turned on is used to transmit process the data, and display instantaneous torque and angle (more on this in the following section). A rechargeable battery (P0YY162WW, Vistaprint, Massachusetts USA) is 2 cm both wide and tall by 9.5 cm long. Its capacity is 2200 mAh and it can provide a maximum current of 1 amp. Ribbon wires with 15 wires in a strip, each 1.3 mm diameter are used.

User Interface and Software. The in-the-loop graphical interface is designed by using Arduino and JAVA Eclipse IDEs (version 4.5.2). A sequence of screenshots are shown from opening the app until finishing the recording session. Fig. 3, Panel 6 shows an example session. In Panel 6, the white region on each instantaneous sensor readout indicates the ideal region. Therefore, a torque larger than 30 Nm or angle less than 13 degrees gives a rough indication of a stiff knee. For the case shown, the torque for the previous session was 3 Nm while the maximum torque so far in this session is 13 Nm and instantaneously is 7 Nm. Therefore, the knee showed a lower maximum torque value in the prior session. On the other

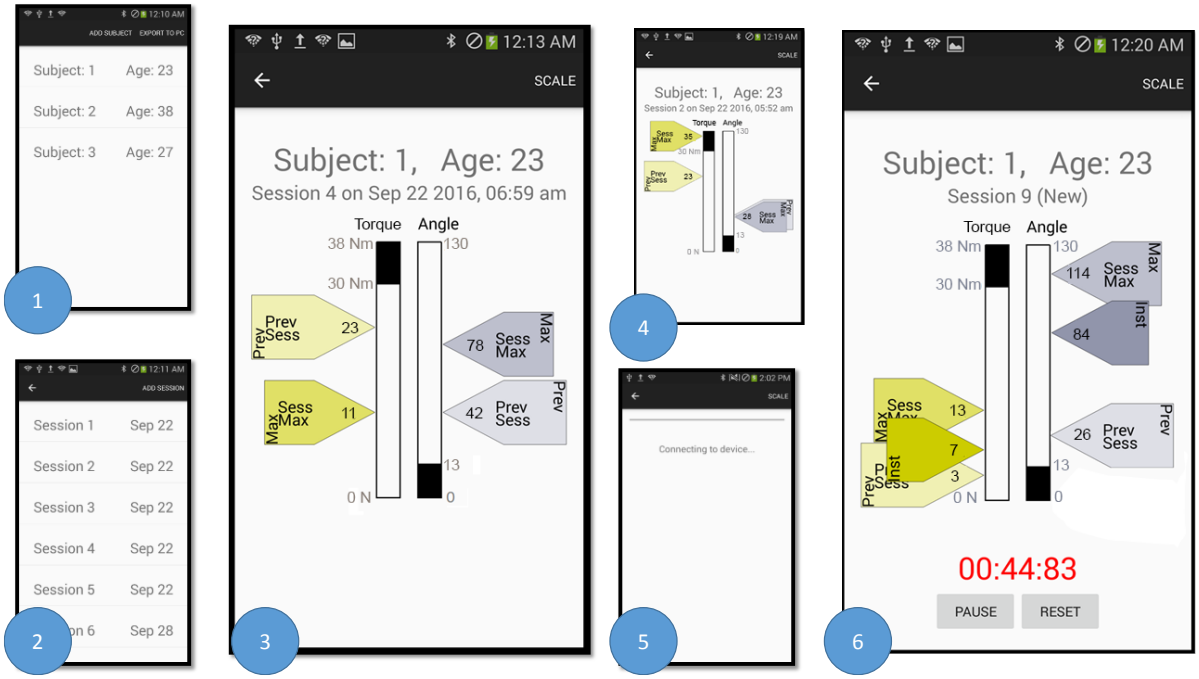

Fig 3. Sequence of user steps on the mobile phone app. Step 1 is to add, delete, edit, or select a patient. Step 2 is to add or view a session for that patient. If a prior session is selected, Step 3 shows its data. Step 4 is a display of data from a prior session. Step 5 is what shows up if a new session is created in Step 2 and the app needs to connect to the Arduino. Step 6 shows the instantaneous values of an active recording session.

hand, the maximum angle of the current session (114 degrees) is more than the previous session (26 degrees). The physician is moving to a greater range of motion.

Mobile phone and Battery. The mobile phone (Samsung Galaxy S4 with Android operator system GT-I9500, model, Samsung, Seoul, Korea) with Bluetooth turned on can be a transmitter, processing the data, and displaying the results (more on this in the following section). A rechargeable battery (P0YY162WW, Vistaprint, Massachusetts USA) is 2 cm both wide and tall by 9.5 cm long. Its capacity is 2200 mAh and it can provide a maximum current of 1 amp. Ribbon wires with 15 wires in a strip, each 1.3 mm diameter, make the design compact.

User Interface and Software. The in-the-loop interface is designed by using Arduino and JAVA Eclipse IDEs (version 4.5.2). A sequence of screenshots are shown from opening the app until finishing the recording session. Fig. 3, Panel 6 shows a current recording session. In Panel 6, the white

region on each instantaneous sensor readout indicates the ideal region. Therefore, a torque bigger than 30 Nm or angle less than 13 degrees gives a rough indication of a stiff knee. For the subject shown the torque for the previous session was 3 Nm while the maximum torque so far in this session is 13 Nm and instantaneously is 7 Nm. Therefore, the knee showed a lower maximum torque value in the prior sesion. On the other hand, the maximum angle of the current session (114 degrees) is more than the previous session (26 degrees). This conveys that the physician is moving the knee to a greater range of motion.

3. EXPERIMENTS

a. CALIBRATION TESTS

Both force and angle were calibrated with the entire device in place. Regarding force, we used a series of gym plate masses from 2.26 to 18 kg, freely resting on a 3D printed flat fixture that was positioned onto the u-shaped hand fixture. At the time the brace was affixed to a healthy participant with his leg at an angle of 0 degrees. The reading from the two load cells added together was compared to that of the masses. Three replications in measurement were performed for each mass. The results indicated that a maximum force of 177 N could be exerted and the mapping of the actual load to the torque readout from the sensors was quite linear with an R-squared value of 0.98, Fig. 4. Equation (1) gives the conversion between the torque reading and calibrated value. Note that force was turned into torque by multiplying by a lever arm of 21 cm.

$$\text{Meas.force} = -0.59 + 1.32 \times \text{sensor reading} \quad (1)$$

Regarding rotational angle, the voltage output of the potentiometer was evaluated to attain a mapping to the angle of upper and lower leg as measured by a goniometer that is a built in part of the knee brace. Fig. 5 shows that the fit of a quadratic equation yielded an R-squared value of 0.99, and relationship as described by Equation (3).

$$\text{Meas.angle} = 13.18 \times voltage^2 + 3.4 \times voltage - 3.42 \quad (2)$$

b. HUMAN SUBJECTS EXPERIMENTS IN THE CLINIC

Participants. To preliminarily evaluate the device, two patients who were to undergo knee manipulation were recruited for the study. The two male patients were of age 76 who had knee stiffness but no prior knee manipulation, and age 34 who had had a knee scar removed previously through a knee manipulation. Approval to conduct the study had been granted from the Internal Review Board (IRB) at the University of Virginia.

Experimental procedures. Informed consent was obtained at a clinical visit prior to the knee manipulation procedure. Then, each patient underwent the standard procedure for knee manipulation under anesthesia. The brace was fitted once the patient was under anesthesia, whereby an orthopedic physician slid the leg through the straps of the brace, and each strap was tightened and secured. The physician then exerted force by pressing on the u-shape plate and otherwise conducting the procedure according to that described in Section II.A. The physician occasionally looked at the user interface to observe the current force and angle, but the majority of his focus was upon the patient's leg and knee. Once the manipulation was complete, the device was removed and the typical plan of care continued. Each session with a patient was about 15-20 minutes long. The actual manipulation procedure took place in less than 1-2 minutes.

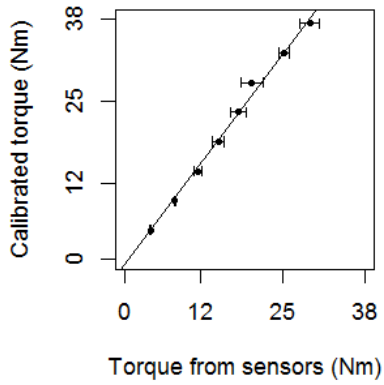

Fig 4. Torque read from the two load cells added together versus that calibrated with masses of known values.

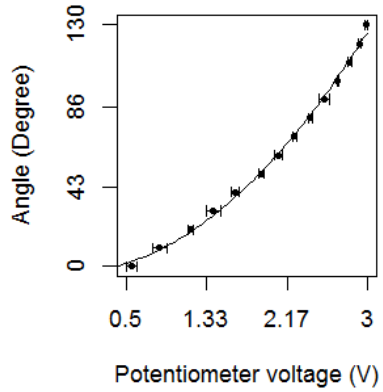

Fig 5. Potentiometer voltage output versus goniometer measured angle.

Participants. Two patients who were to undergo knee manipulation were recruited for the study. The patients were of age 76 who had knee stiffness, and age 34 who had had a knee scar removed. Approvals to conduct the study had been granted from the Internal Review Board (IRB) at the University of Virginia.

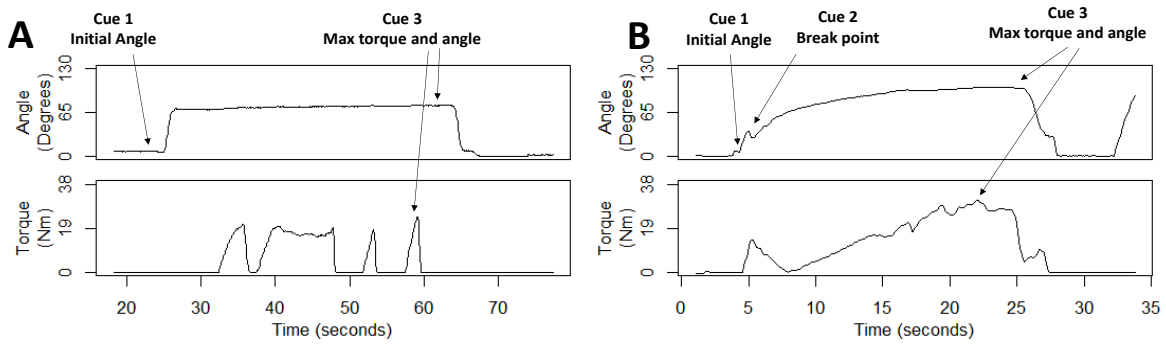

Fig 6. Angle and torque measurements over time for the knee manipulations of participants 1 (Panel A) and 2 (Panel B).

Experimental procedures. Informed consent was done in the clinic on a visit prior to the knee manipulation procedure. On a subsequent day, each patient underwent the standard procedure for knee manipulation under anesthesia. The brace was fitted once the patient was under anesthesia, whereby an orthopedist slide the leg through the straps of the brace, and each strap was tightened and secured. The physician then exerted force by pressing on the u-shape plate and otherwise conducting the procedure according to that described in Section II.A. The physician occasionally looked at the user interface to see what the current force and angle were, but the majority of focus was upon the patient's leg. Once the manipulation was complete, the brace was removed and the usual clinical care continued. Each session with a patient was about 15-20 minutes long. The actual manipulation procedure took place in less than 1-2 minutes.

4. RESULTS

Overall, the results of the human subjects experiments indicated that the maximum torque observed was 31 Nm (or 147 N, within the upper limit of 177 N achievable) and the range of motion spanned 100 degrees (also within the device's upper limit of 130 degrees).

For participant 1, the knee presented an initial resting position of 8 degrees, Fig. 6A and Table 1. At around 26 seconds, the physician moved the knee to an angle of 77 degrees, without his hand on the u-shaped force exertion fixture. At about 37 seconds, the physician exerted torque on the knee of about 23.6 Nm. Then, the physician stopped exerting force before re-exerting force three more times. Note that the angle did not move even as torque up to about 23.6 Nm was applied at 59 seconds. The experimenters observed that the knee in this case was so stiff that even when greater magnitudes of torque were exerted, the observable angle of the knee did not noticeably change. During the procedure, the physician made a judgment that the knee should not be delivered a greater magnitude of torque. This judgment was based on his perception of torque observed at that maximum angle, as well as the patient's history. Therefore, scar tissue was not broken, and the medical team decided to end the manipulation and perform an arthroscopic lysis of adhesions.

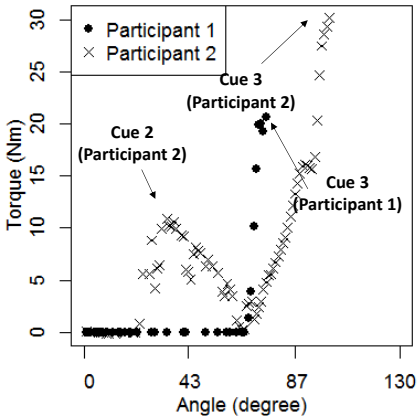

Fig 7. Torque over range of motion for the two participants

For participant 2, the knee presented an initial resting position of 9 degrees. Recall this participant had had a knee manipulation previously. Therefore, the state of knee upon beginning the procedure was less stiff than that of the prior participant. As observable in Fig. 6B, his knee reached 100 degrees of flexion at 20 seconds and the scar tissue was broken free at 6 seconds. The topic of how we determined this is the subject of the following paragraph. The maximum torque was 31 Nm (or 147 N, within the upper limit of 177 N achievable). For comparison, Dong, et al. used torque values smaller than

80 Nm [12], Lane found passive knee moment ranged from 3.5 to 53 Nm [14] and for reference about 130 Nm is required to tighten a tire's lug nut.

Further analysis was done to compare both participants, in terms of knee angle as a function of torque, Fig. 7. Note that in plotting the data angle was rounded down to an integer value and then the maximum torque at each integer angle, over the whole duration of each experiment was plotted. This procedure was utilized to remove time as a factor, which can differ significantly as a function of user induced variability in how a physician moves and hold in place in performing the procedure.

For participant 1 (who had a stiff knee), the point of maximum torque at a lesser angle of about 77 degrees as compared to participant 2 (whose the scar was broken through at about 38 degrees) and then reached a point of maximum torque at about 100 degrees. This indicates that the knee of participant 1 was stiffer than that of participant 2. Furthermore, it seems likely that the device can report the break point of scar tissue by the increase in stiffness at about 38 degrees whereby the torque dramatically drops before climbing again as the angle reaches about 100 degrees. In this case, the knee manipulation has loosened up the knee and restored it to a state much closer to the 130 degrees desired by the physician and well over the 90 degree point where issues are noted regarding one's ability to ambulate and perform activities of daily living [3].

The torque values read from the two load cells in parallel are added together. Ideally, they should be equal, but in practice the physician may not have his hand centered on the u-shaped bracket. Fig. 8 compares the two traces. While the absolute magnitude difference between the two traces lies within (0.33, 0.73) Nm with 95% confidence, in a paired t-test, the torque values do visually differ between the two at maximum torque.

Table 1. Summary of psychomotor cues for two participants

	Participant 1	Participant 2
Cue 1	8 degrees	9 degrees
Cue 2	N/A	38 degrees, 14 Nm
Cue 3	77 degrees , 23.6 Nm	100 degrees, 31 Nm

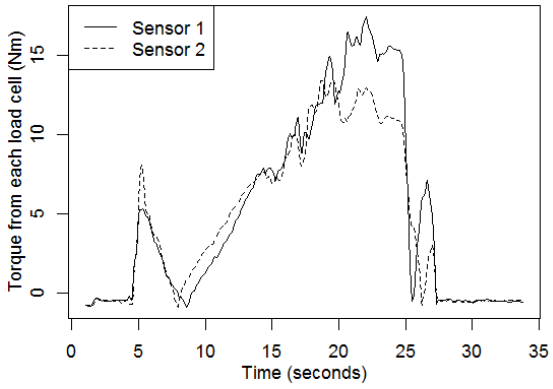

Fig 8. Discrepancy between the readings of two force sensors for participant 2's session

5. DISCUSSION

This work herein created a proof-of-concept prototype for the orthopedic evaluation of knee stiffness in knee arthrofibrosis patients. Preliminary tests with two participants undergoing knee manipulation indicate that physicians tend to exert 31 Nm of torque on the knee and position the upper and lower leg to angles of 100 degrees. These values are within the capability of the device and those reported in the literature [11], [12], [14]. In terms of the three psychomotor cues observed, this work shows that they can be quantified, and may be of use in differentiating two knee conditions and severity.

Several considerations and issues remain to be addressed in evaluating the device. First, the study only included two participants, and while the force and angle ranges fit within the device's requirements a broader cohort is needed. Second, the prototype shares common characteristics with the Smart Knee device [11]. In particular, this device's theoretical accuracy is less than a degree, but due to consistency issues in fitting the device to a patient's leg, the accuracy is 8 degrees in practice. Likewise, our device can theoretically detect 0.55 degree changes in angle, but we have not tested what further error might be introduced by variations in device installation on a leg. Third, as this device is physician controlled, whomever evaluates the data must account for a physician taking her or his hand off the brace

versus torque decreases for reasons related to the knee's condition. We attempt to do this by removing time an analyzing torque at the first or maximum angle encountered, but more work is needed here to test the variability induced by individual physicians. Fourth, the determination of the breakpoint needs to be more rigorously demonstrated. The concept is demonstrated here, but only for one participant. The main concept is that torque drops so that much less is required for every unit increase in angle, until reaching the maximum range of motion where torque again increases. This concept needs to be algorithmically defined, and clearly differentiated from the case where the physician removes his hand from the u-shaped fixture. In addition, such an algorithm would have to account for the possibility that there could be multiple breakpoints that come as a result of breaking of scar tissue at particular joint capsules. Fifth, we note that the physician only gets one chance to manipulate a knee. So even if there are 7-8 manipulation movements on a particular day, the first is unique from the next 6-7. This presents the opportunity to compare between the first and latter manipulations, something not done here, we focused on maximum torque.

Finally, the focus herein was on the clinical procedure of MUA as done under anesthesia. In the future, the device may also be useful in evaluating the knee before an MUA, as a way to determine which patients are most eligible for this procedure as opposed to alternative treatments.

6. ACKNOWLEDGMENT

This work was not supported by any grants.

7. REFERENCE

[1] M. G. Wilson, K. Kelley, and T. S. Thornhill, "Infection as a complication of total knee-replacement arthroplasty. Risk factors and treatment in sixty-seven cases," J Bone Joint Surg Am, vol. 72, no. 6, pp. 878–883, Jul. 1990.

[2] J. P. Furia and V. D. Pellegrini, "Heterotopic ossification following primary total knee arthroplasty," J Arthroplasty, vol. 10, no. 4, pp. 413–419, Aug. 1995.

[3] D. Magit, A. Wolff, K. Sutton, and M. J. Medvecky, "Arthrofibrosis of the Knee," J Am Acad Orthop Surg, vol. 15, no. 11, pp. 682–694, Nov. 2007.

[4] K. N. Laubenthal, G. L. Smidt, and D. B. Kettelkamp, "A quantitative analysis of knee motion during activities of daily living," Phys Ther, vol. 52, no. 1, pp. 34–43, Jan. 1972.

[5] A. A. Sassoon, O. O. Adigweme, J. Langford, K. J. Koval, and G. J. Haidukewych, "Manipulation Under Anesthesia: A Safe and Effective Treatment for Posttraumatic Arthrofibrosis of the Knee," J Orthop Trauma, vol. 29, no. 12, pp. e464-468, Dec. 2015.

[6] J. L. Fox and R. Poss, "The role of manipulation following total knee replacement," J Bone Joint Surg Am, vol. 63, no. 3, pp. 357–362, Mar. 1981.

[7] I. Ipach, R. Schäfer, J. Lahrmann, and T. Kluba, "Stiffness after knee arthrotomy: evaluation of prevalence and results after manipulation under anaesthesia," Orthop Traumatol Surg Res, vol. 97, no. 3, pp. 292–296, May 2011.

[8] S. E. Fitzsimmons, E. A. Vazquez, and M. J. Bronson, "How to treat the stiff total knee arthroplasty: a systematic review," Clin. Orthop. Relat. Res., vol. 468, no. 4, pp. 1096–1106, Apr. 2010.

[9] H. Ghani, N. Maffulli, and V. Khanduja, "Management of stiffness following total knee arthroplasty: A systematic review," The Knee, vol. 19, no. 6, pp. 751–759, Dec. 2012.

[10] E. M. Keating, M. A. Ritter, L. D. Harty, G. Haas, J. B. Meding, P. M. Faris, and M. E. Berend, "Manipulation after total knee arthroplasty," J Bone Joint Surg Am, vol. 89, no. 2, pp. 282–286, Feb. 2007.

[11] "Bend • Intelligent knee brace platform for healthcare." [Online]. Available: http://www.bendlabs.com/smartknee/. [Accessed: 27-Dec-2016].

[12] S. Dong, K.-Q. Lu, J. Q. Sun, and K. Rudolph, "Smart Rehabilitation Devices: Part I – Force Tracking Control," J Intell Mater Syst Struct, vol. 17, no. 6, pp. 543–552, 2006.

[13] M. R. Mahfouz, R. C. Wasielewski, and R. Komistek, "Noninvasive diagnostic system," US 20120029345 A1, 02-Feb-2012.

[14] J. Lane, "Knee joint stiffness and function following Total Knee Arthroplasty," The University of Edinburgh, 2010.

APPENDIX

1. 3D PERSPECTIVE VIEWS OF ALL COMPONENTS

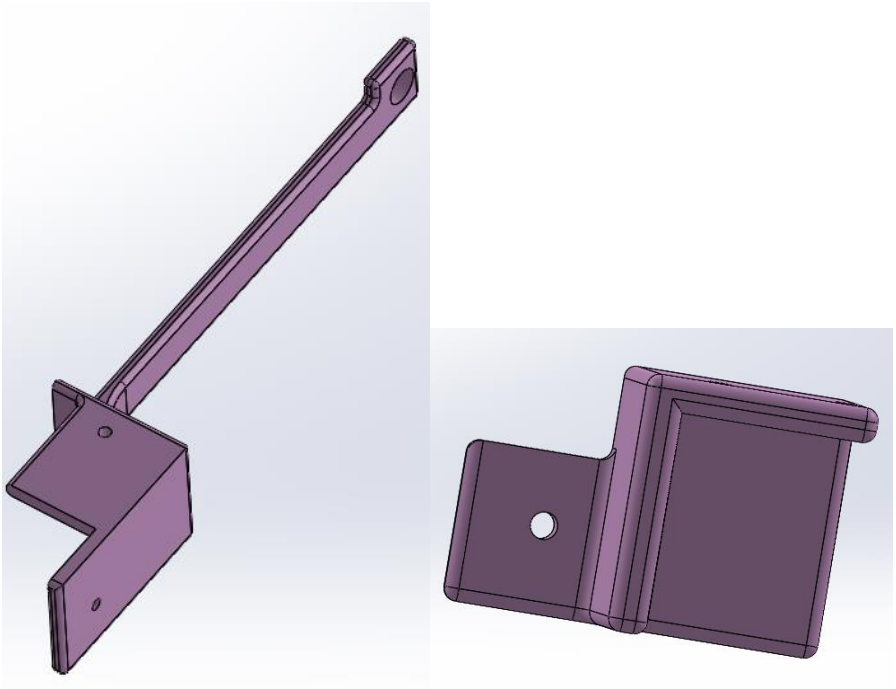

Fig 9. Side parts

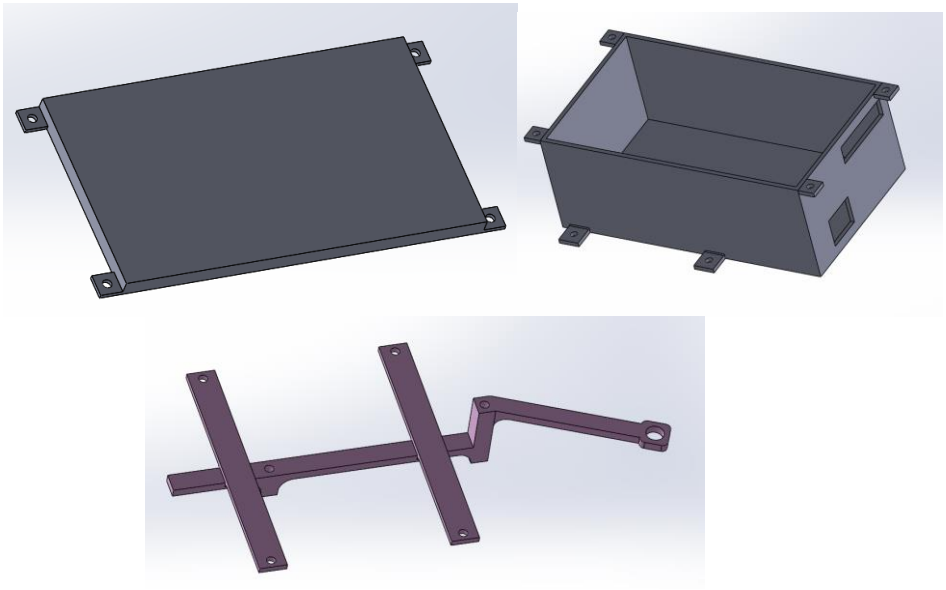

Fig 10. Box to put electrical components

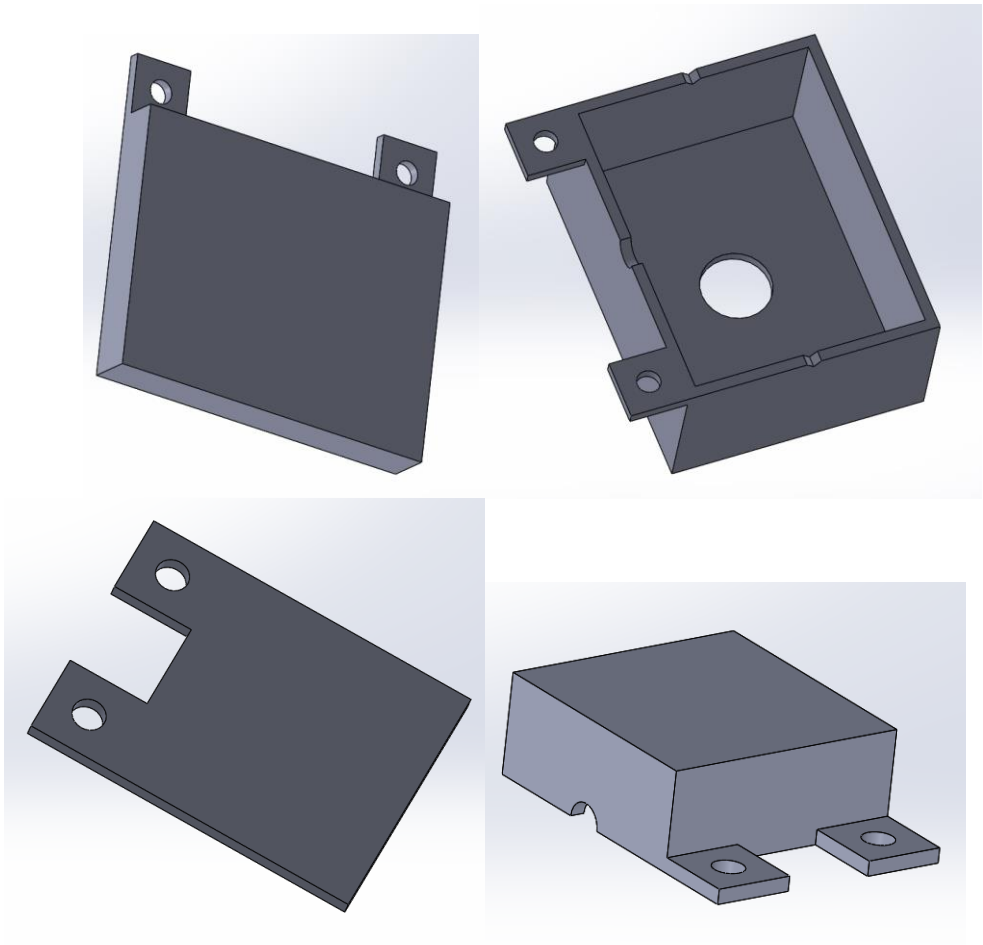

Fig 11. Box to contain wires and connectors.

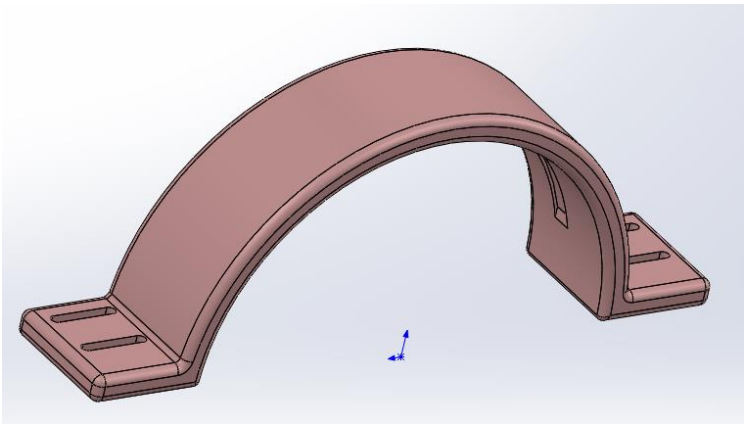

Fig 12. This part holds the doctor's hand.

Fig 9 has the following from left to right:

1) This part connects the potentiometer to the electrical box.

2) This part connects one load cell to the potentiometer.

3) This part connects the two load cells and the place for the doctor to push against.

2. CIRCUIT DIAGRAMS

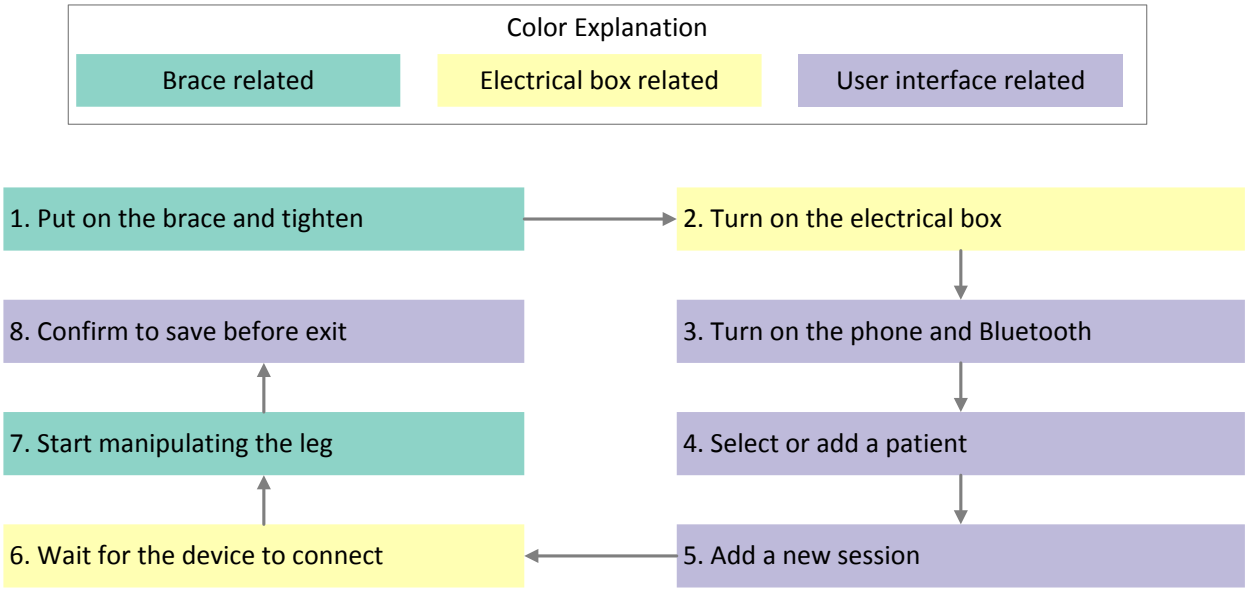

Fig 13. Work flow

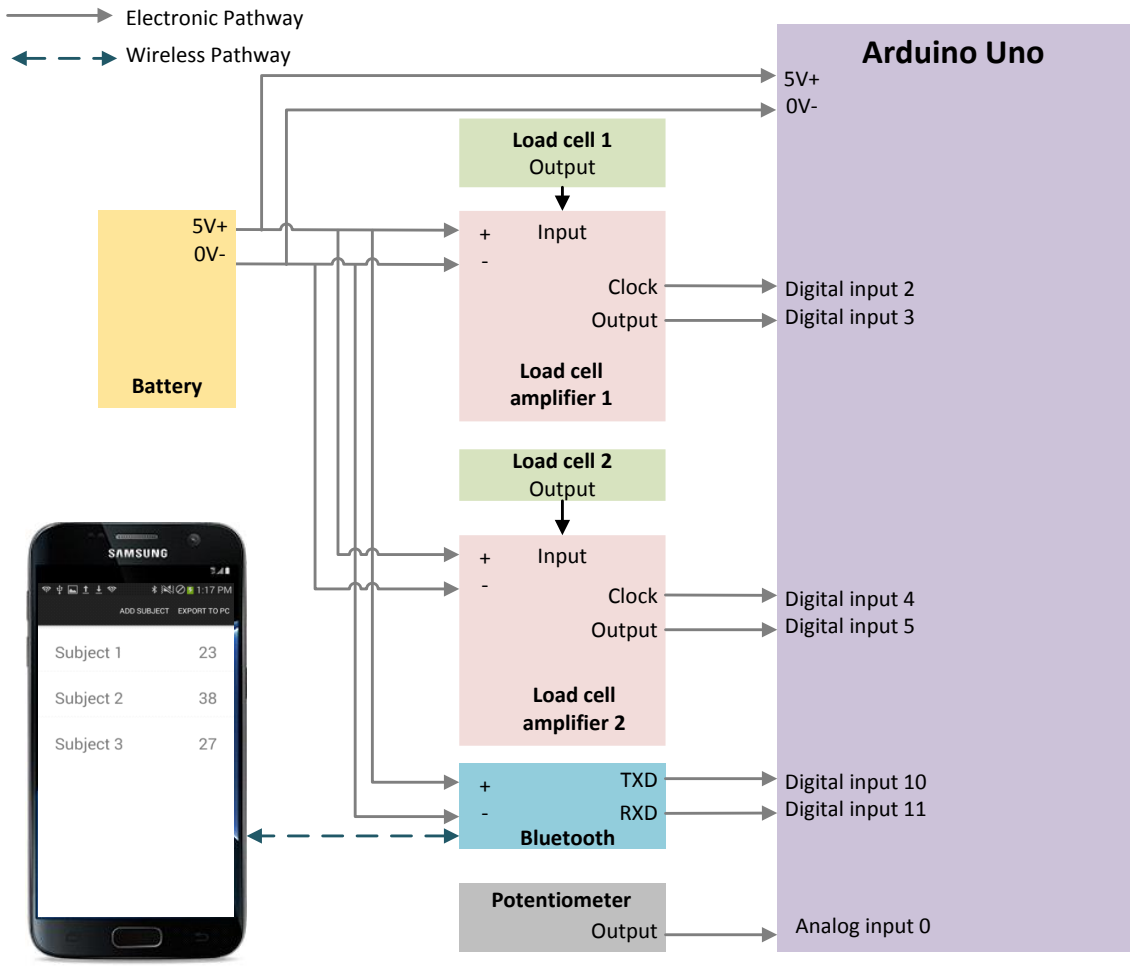

Fig 14. High level view

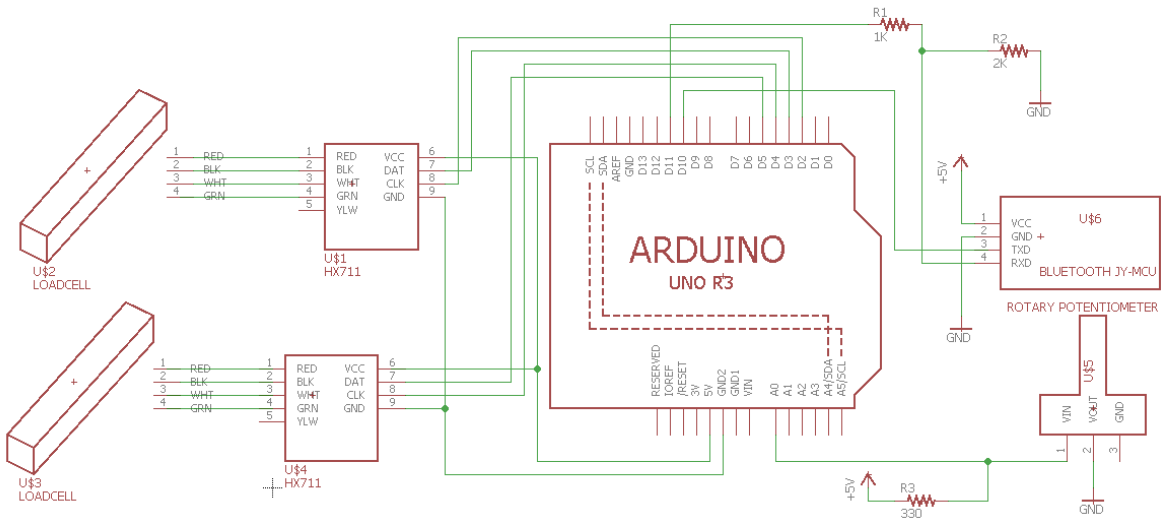

Fig 15. Schematic view

3. Code Line By Line

a. Code For Microcontroller

```
#include <SoftwareSerial.h>

const int RX_PIN = 10;
const int TX_PIN = 11;
const int BLUETOOTH_BAUD_RATE = 9600;


#include "HX711.h"

#define DOUT1  3
#define CLK1  2

#define DOUT2  5
#define CLK2  4

HX711 scale1(DOUT1, CLK1);
HX711 scale2(DOUT2, CLK2);

float calibration_factor = 7050; //-7050 worked for my 440lb max scale setup


SoftwareSerial bluetooth(RX_PIN, TX_PIN);
long randomNumber;
float maxVol3=-1;
float minVol3=10;
void setup() {
   Serial.begin(9600);
   bluetooth.begin(BLUETOOTH_BAUD_RATE);
  //Serial.print("asdsda");
  scale1.set_scale();scale2.set_scale();
  scale1.tare();  scale2.tare();
  long zero_factor = scale1.read_average(); //Get a baseline reading
  //Serial.print("Zero factor: ");
  //Serial.println(zero_factor);
  scale1.set_scale(calibration_factor); //Adjust to this calibration factor
  scale2.set_scale(calibration_factor);
}
long counter=0;
void loop() {

  float angle= analogRead(A0)     *(5.0/1023.0);
  //Serial.println(angle);
```

```
//  Serial.print(" ");
//  Serial.println(vol2);
  counter=counter+1;
  if (counter>1000) counter=0;
  //bluetooth.print(counter); bluetooth.print(" ");
  float sca1=scale1.get_units();
  float sca2=scale2.get_units();
  //
//  Serial.print(sca1,4);Serial.print(" ");
//  Serial.println(sca2,4);
  bluetooth.print(sca1, 4);  bluetooth.print(" ");
  bluetooth.print(sca2, 4);  bluetooth.print(" ");
  bluetooth.print(angle, 4);
  //Serial.println(angle,4);
  if (maxVol3<angle) maxVol3=angle;
  if (minVol3>angle) minVol3=angle;
  //Serial.print(maxVol3); Serial.print(" ");Serial.println(minVol3);
//  Serial.println(vol3);
  bluetooth.println();
  //if (Serial.available()) {
  // char temp = Serial.read();
  //  if(temp == '+' )
  //    scale.tare();

  //}
  if (bluetooth.available()) {
    char temp = bluetooth.read();
    if(temp == '+' )
      scale1.tare();

  }
  delay(50);


}
```

b. CODE FOR THE ANDROID APP

cPoint.java

```
package com.example.com.hdl.first_tutorial;

public class cPoint{
        public float x,y;
        public cPoint(float xx,float yy){
                x=xx;
```

```
            y=yy;
        }
}
```

MainActivity.java

```
package com.example.com.hdl.first_tutorial;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.PrintWriter;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Iterator;
import java.util.Scanner;
import java.util.Set;

import com.example.com.hdl.first_tutorial.extraClass.Subject;
import com.example.com.hdl.first_tutorial.extraClass.someMethod;
import com.example.com.hdl.first_tutorial.extraClass.visitInfo;

import android.annotation.SuppressLint;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Intent;
import android.database.sqlite.SQLiteDatabase;
import android.graphics.Color;
import android.media.AudioManager;
import android.media.ToneGenerator;
import android.os.Bundle;
import android.os.Environment;
import android.os.Handler;
import android.os.ParcelUuid;
import android.os.SystemClock;
import android.support.v7.app.ActionBar;
import android.support.v7.app.ActionBarActivity;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
```

```
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;

@SuppressLint("SimpleDateFormat")
@SuppressWarnings("deprecation")
public class MainActivity extends ActionBarActivity {
	Button record, reset;
	TextView mainView, time;

	OutputStream outputStream;
	InputStream inStream;
	BluetoothSocket socket;
	Scanner source;
	PrintWriter print;
	float minV = 2.56f;//2.43f;//2.94f;
	float maxV = 0f;
	float lastAngleRaw;
	float lastAvgF;
	boolean recording = false;
	Handler mHandler;
	ProgressBar progress;

	public void showProgress(int num) {
		progress.setProgress(num);
	}

	boolean simulated = true;

	private void init() throws Exception {

		BluetoothAdapter blueAdapter = BluetoothAdapter.getDefaultAdapter();
		showProgress(20);
		if (blueAdapter != null) {
			if (blueAdapter.isEnabled()) {
				// blueAdapter.cancelDiscovery();
				Set<BluetoothDevice> bondedDevices =
blueAdapter.getBondedDevices();
				showProgress(40);
				if (bondedDevices.size() > 0) {
					try {
						BluetoothDevice device = null;
						Iterator<BluetoothDevice> it =
bondedDevices.iterator();
						while (it.hasNext()) {
							BluetoothDevice next = it.next();
```

```
                                        if (next.getName().equals("HC-06")) {
                                                device = next;
                                                break;
                                        }
                                }

                                if (device == null)
                                        throw new IOException();
                                ParcelUuid[] uuids = device.getUuids();
                                socket =
device.createRfcommSocketToServiceRecord(uuids[0].getUuid());
                                showProgress(60);
                                socket.connect();
                                showProgress(90);
                                outputStream = socket.getOutputStream();
                                inStream = socket.getInputStream();
                                showProgress(100);
                        } catch (IOException e) {
                                throw new Exception("The knee might be off or
out of battery. Please turn on");

                        }

                } else

                        throw new Exception("There is no pair devices");

        } else {

                throw new Exception("Bluetooth is disabled. Please enable
Bluetooth");
        }
    }

    }

    void appendMessage(String a) {
        mainView.setText(a);

    }

    public void write(String s) throws IOException {
        outputStream.write(s.getBytes());
    }

    public static float clamp(float val, float min, float max) {
```

```
            return Math.max(min, Math.min(max, val));
        }

        class Initializetask implements Runnable {
            @Override
            public void run() {

                try {
                    init();
                    MainActivity.this.runOnUiThread(new Runnable() {
                        public void run() {
                            setConnected(true);
                        }
                    });

                    runData();
                } catch (Exception e) {
                    helper.showMessage(mHandler, MainActivity.this, e.getMessage(),
new someMethod() {

                        @Override
                        public void run() {
                            MainActivity.this.finish();
                        }
                    });

                }

            }

        }

        void runData() {

            try {
                source = new Scanner(inStream);
                source.useDelimiter("\n");
                while (true && inStream != null) {
                    // bytes = inStream.read(buffer, bytes, BUFFER_SIZE - bytes);

                    final String tam = source.next();
                    final String[] parts = tam.split(" ");

                    if (parts.length == 3) {
                        final float mot = Math.abs(Float.parseFloat(parts[0]));
                        final float hai = Math.abs(Float.parseFloat(parts[1]));
```

```
                                    final float fullF = mot + hai;
                                    lastAngleRaw = Float.parseFloat(parts[2]);
                                    // appendMessage(String.format("%.3f lbs- %.3f-
                                    // %.3f lbs", mot, hai, avgF));

                                    // float minF = 0.64f;
                                    // float maxF = 3.84f;
                                    float angle = clamp((lastAngleRaw - minV) / (maxV -
minV), 0, 1);
                                    final float realAngle = (angleHB.max - angleHB.min) * (1
- angle) + angleHB.min;

                                    mHandler.post(new Runnable() {
                                        @Override
                                        public final void run() {

        forceHB.setVal(fullF,"");//String.format("%.2f", mot));

        angleHB.setVal(realAngle,"");//String.format("%.2f", lastAngleRaw));
                                                if (recording)
                                                    print.printf("%s %.4f %.4f %.4f\n",
String.format("%2d:%02d:%03d", mins, secs, milliseconds),
                                                            realAngle,mot, hai );
                                                if (forceThreshold > lastAvgF && fullF >
forceThreshold) {
                                                    ToneGenerator toneGen1 = new
ToneGenerator(AudioManager.STREAM_MUSIC, 100);

        toneGen1.startTone(ToneGenerator.TONE_CDMA_PIP, 150);
                                                }
                                            }
                                    });

                                    lastAvgF = fullF;

                                }

                                try {
                                    Thread.sleep(50);
                                } catch (InterruptedException e) {
                                    e.printStackTrace();
                                }
                            } // end while
                        } catch (NumberFormatException e) {
                            e.printStackTrace();
                        } catch (Exception e) {
```

```
                    showError(e);
                }
        }// end void run

        void showError(Exception e) {
                helper.showMessage(mHandler, MainActivity.this, e.getMessage(), new
someMethod() {

                    @Override
                    public void run() {
                        MainActivity.this.finish();
                    }
                });
        }

        long starttime = 0L;
        long timeInMilliseconds = 0L;
        long timeSwapBuff = 0L;
        long updatedtime = 0L;
        int secs = 0;
        int mins = 0;
        int milliseconds = 0;
        Handler handler = new Handler();
        public Runnable updateTimer = new Runnable() {

                public void run() {

                    timeInMilliseconds = SystemClock.uptimeMillis() - starttime;

                    updatedtime = timeSwapBuff + timeInMilliseconds;

                    secs = (int) (updatedtime / 1000);
                    mins = secs / 60;
                    secs = secs % 60;
                    milliseconds = (int) (updatedtime % 1000);
                    time.setText(String.format("%02d:%02d:%02d", mins, secs,
milliseconds/10));

                    handler.postDelayed(this, 0);
                }

        };

        public void startRec(View v) {
                if (!recording) {// if not recording
                    time.setTextColor(Color.RED);
```

```
                record.setText("Pause");
                recording = true;
                starttime = SystemClock.uptimeMillis();
                handler.postDelayed(updateTimer, 0);
            } else {// if recording
                stopRecording();
            }

        }

        void stopRecording() {
            time.setTextColor(Color.BLACK);
            record.setText("Record");
            recording = false;
            timeSwapBuff += timeInMilliseconds;
            handler.removeCallbacks(updateTimer);
        }

        public void resetRec(View v) {
            starttime = 0L;
            timeInMilliseconds = 0L;
            timeSwapBuff = 0L;
            updatedtime = 0L;
            recording = false;
            secs = 0;
            mins = 0;
            milliseconds = 0;
            record.setText("Record");
            handler.removeCallbacks(updateTimer);
            time.setText("00:00:00");
            angleHB.currentMax = 0;
            forceHB.currentMax = 0;
            time.setTextColor(Color.BLACK);
            reOpenFile();
        }

        File getFile() {
            return new File(Environment.getExternalStorageDirectory().toString(),
"arun1.txt");
        }

        void reOpenFile() {
            File file = getFile();
            if (print != null)
                print.close();
            try {
```

```
                print = new PrintWriter(file);
            } catch (FileNotFoundException e) {
                e.printStackTrace();
            }
        }

        public void taringScale(View v) {
            try {
                write("+");
            } catch (IOException e) {
                Toast.makeText(getApplicationContext(), "Cannot tare",
Toast.LENGTH_SHORT).show();
            }
        }

        public void resetAngle(View v) {
            maxV = lastAngleRaw;
        }

        public void disconnectBluetooth() {

            if (inStream != null) {
                try {
                    inStream.close();
                } catch (Exception e) {
                }
                inStream = null;
            }
            appendMessage("In stream gone\n");
            if (outputStream != null) {
                try {
                    outputStream.close();
                } catch (Exception e) {
                }
                outputStream = null;
            }
            appendMessage("Out Stream gone\n");
            if (socket != null) {
                try {
                    socket.close();
                } catch (Exception e) {
                }
                socket = null;
            }
            appendMessage("Socket closed\n");
            appendMessage("Disconnected\n");
```

```
        }

        float forceThreshold;
        heightBar forceHB, angleHB;
        SQLiteDatabase lite;

        visitInfo sess = null;
        Subject sub = null;

        float[] cc(float[] arr, float con) {
                float tm = arr[2] * con;
                return new float[] { arr[0], arr[1], tm };
        }

        int darker(int[] base, float percent) {

                int t = percent < 0 ? 0 : 255;
                float p = percent < 0 ? percent * -1 : percent;
               return Color.rgb(Math.round((t - base[0]) * p) + base[0], Math.round((t - base[1])
* p) + base[1],
                                Math.round((t - base[2]) * p) + base[2]);
        }

        TextView name;

        @Override
        protected void onCreate(Bundle savedInstanceState) {

                super.onCreate(savedInstanceState);

                setContentView(R.layout.activity_main);

                progress = (ProgressBar) findViewById(R.id.progressF);
                forceHB = (heightBar) findViewById(R.id.forceHB);
                angleHB = (heightBar) findViewById(R.id.angleHB);
                time = (TextView) findViewById(R.id.timeRec);
                record = (Button) findViewById(R.id.record);
                reset = (Button) findViewById(R.id.reset);
                name = (TextView) findViewById(R.id.description);
                mainView = (TextView) findViewById(R.id.message);

                mHandler = new Handler();
                Intent i = getIntent();

                if (i != null) {
```

```
                sess = (visitInfo) i.getSerializableExtra("visitData");
                sub = (Subject) i.getSerializableExtra("subData");
                lite = helper.openDatabase(this);
                name.setText(String.format("Subject: %s,   Age: %d", sub.pseudo,
sub.age));
        }

        int[] base1 = new int[] { 204, 204, 0 };
        int[] base2 = new int[] { 145, 149, 171 };
        forceHB.setColor(Color.BLACK, Color.YELLOW, darker(base1, 0),
darker(base1, 0.7f), darker(base1, 0.4f));
        angleHB.setColor(Color.BLACK, Color.YELLOW, darker(base2, 0),
darker(base2, 0.7f), darker(base2, 0.4f));

        // Color.rgb(145,149,171),
        // Color.rgb(189,193,218), Color.rgb(182,187,214));
        forceHB.setLabel("Force");
        angleHB.setLabel("Angle");

        forceHB.updateMinMaxPrecision(0f, 2000f, 0);
        angleHB.updateMinMaxPrecision(0f, 100, 0);

        forceHB.updateThresholdAndFlip(1500, false);
        angleHB.updateThresholdAndFlip(20, true);

        time.setTextColor(Color.BLACK);
        boolean self = false;

        if (sess.visitId != -1) {
                appendMessage(String.format("Session %d on %s", sess.visitId,
                                new SimpleDateFormat("MMM dd yyyy, hh:mm
a").format(sess.visitTime)
                                .replace("AM", "am").replace("PM","pm")));
                simulated = true;
                forceHB.setVisibleInst(false);
                angleHB.setVisibleInst(false);
                if (!self) {
                        forceHB.setMax(sess.maxForce);
                        angleHB.setMax(sess.maxAngle);
                } else {
                        forceHB.setMax(115);
                        angleHB.setMax(43);
                }
                time.setVisibility(View.INVISIBLE);
                record.setVisibility(View.INVISIBLE);
                reset.setVisibility(View.INVISIBLE);
```

```
                progress.setVisibility(View.INVISIBLE);
            } else { // adding new
                sess.visitId = (Integer) helper.getMaxID(lite,
                        "select max(visitid) from visitinfo where subjectid=" +
sub.subjectID, false) + 1;
                setConnected(false);
                simulated = false;
            }
            if (!self) {
                Object preMaxForce = helper.getMaxID(lite,
                        "select maxForce from visitinfo where subjectid=" +
sub.subjectID + " and visitid<" + sess.visitId,
                        true);
                if (preMaxForce == null) {
                    forceHB.setVisiblePre(false);
                    angleHB.setVisiblePre(false);
                } else {
                    forceHB.setPrevious((Float) preMaxForce);
                    angleHB.setPrevious((Float) helper.getMaxID(lite, "select
maxAngle from visitinfo where subjectid="
                            + sub.subjectID + " and visitid<" + sess.visitId,
true));
                }
            } else {
                forceHB.setPrevious(75);
                angleHB.setPrevious(46);
            }
        }

        void setConnected(boolean ok) {
            if (ok) {

                appendMessage(String.format("Session %d (Current)", sess.visitId));
                progress.setVisibility(View.INVISIBLE);
                forceHB.setVisibility(View.VISIBLE);
                angleHB.setVisibility(View.VISIBLE);
                time.setVisibility(View.VISIBLE);
                record.setVisibility(View.VISIBLE);
                reset.setVisibility(View.VISIBLE);
                name.setVisibility(View.VISIBLE);

                forceThreshold = 40f;

                reOpenFile();
                Thread.currentThread().setUncaughtExceptionHandler(new
Thread.UncaughtExceptionHandler() {
```

```
                        public void uncaughtException(Thread thread, Throwable e) {
                                showError((Exception) e);
                        }
                });

        } else {
                progress.setVisibility(View.VISIBLE);
                forceHB.setVisibility(View.INVISIBLE);
                angleHB.setVisibility(View.INVISIBLE);
                time.setVisibility(View.INVISIBLE);
                record.setVisibility(View.INVISIBLE);
                reset.setVisibility(View.INVISIBLE);
                name.setVisibility(View.INVISIBLE);
                appendMessage("Connecting to device...");
                if (1 == 1) {
                        Thread thread = new Thread(new Initializetask());
                        thread.setUncaughtExceptionHandler(new
Thread.UncaughtExceptionHandler() {
                                public void uncaughtException(Thread t, Throwable e) {
                                        showError((Exception) e);
                                }
                        });
                        thread.start();
                        startRec(null);
                }
        }

    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        ActionBar actionBar = getSupportActionBar();
        actionBar.setHomeButtonEnabled(true);
        actionBar.setDisplayHomeAsUpEnabled(true);
        actionBar.setDisplayOptions(0, ActionBar.DISPLAY_SHOW_TITLE);

        return true;
    }

    public void saveWork() {
        try {
                print.close();
```

```
                        if (milliseconds + mins + secs == 0) {
                                Toast.makeText(this, "Nothing to save",
Toast.LENGTH_SHORT).show();
                                return;
                        }

                        String s = new SimpleDateFormat("yyyyMMdd hhmmss a").format(new
Date());

                        // if (curVisit == maxVisit) {
                        // // new

                        lite.execSQL(String.format("insert into visitinfo values (%d, %d,
'%s', %f, %f)", sess.visitId,
                                        sub.subjectID, s, angleHB.currentMax,
forceHB.currentMax));

                        // } else {
                        // // update
                        //
                        // lite.execSQL(String.format(
                        // "update visitinfo set visittime='%s', maxangle=%f, maxforce=%f
                        // where subjectid=%d and visitid=%d",
                        // s, angleHB.currentMax, forceHB.currentMax, sub.subjectID,
                        // curVisit));
                        //
                        // }
                        showProgress(30);
                        lite.execSQL(String.format("delete from Experiment where subjectid=%d
and visitid=%d", sub.subjectID,
                                        sess.visitId));
                        Scanner r;
                        showProgress(60);
                        try {
                                r = new Scanner(getFile());
                                int sampleid = 0;
                                while (r.hasNext()) {
                                        String[] read = r.nextLine().trim().split(" ");
                                        sampleid++;
                                        lite.execSQL(String.format("insert into Experiment
values(%d,%d,%d,'%s',%s,%s,%s)", sampleid,
                                                        sub.subjectID, sess.visitId, read[0], read[1],
read[2], read[3]));

                                }
                        } catch (FileNotFoundException e) {
```

```
                    e.printStackTrace();
                }
                showProgress(100);
            } catch (Exception e) {
                showError(e);
            }
        }

        @Override
        public boolean onOptionsItemSelected(MenuItem item) {
            // Handle action bar item clicks here. The action bar will
            // automatically handle clicks on the Home/Up button, so long
            // as you specify a parent activity in AndroidManifest.xml.
            int id = item.getItemId();
            switch (id) {

            case android.R.id.home:
                if (!simulated) {
                    stopRecording();
                    showProgress(0);
                    disconnectBluetooth();
                    showProgress(20);
                    helper.showYesNo(mHandler, MainActivity.this, "Do you want to
save?", new someMethod() {

                        @Override
                        public void run() {
                            saveWork();
                            Toast.makeText(getApplicationContext(), "Saved!",
Toast.LENGTH_SHORT).show();
                            lite.close();
                            MainActivity.this.finish();
                        }
                    }, new someMethod() {
                        @Override
                        public void run() {
                            lite.close();
                            MainActivity.this.finish();
                        }
                    });

                } else
                    this.finish();

                return true;
```

```
            }
            return super.onOptionsItemSelected(item);
        }
}
```

---

heighBar.java

```
package com.example.com.hdl.first_tutorial;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Paint.Align;
import android.graphics.Rect;
import android.util.AttributeSet;
import android.view.View;

public class heightBar extends View {
        boolean visibleInst = true;
        boolean visiblePre = true;

        public void setVisibleInst(boolean yes) {
                visibleInst = yes;
                postInvalidate();
        }

        public void setVisiblePre(boolean yes) {
                visiblePre = yes;
                postInvalidate();
        }
        String extra_tmp="";

        public void setVal(float val, String tmp) {
                if (val > max)
                        val = max;
                if (val < min)
                        val = min;
                currentVal = val;
                if (currentVal > currentMax)
                        currentMax = currentVal;
                extra_tmp=tmp;
                postInvalidate();
        }

        public void setMax(float max) {
```

```
        currentMax = max;
        postInvalidate();
}

float rawPre;

public void setPrevious(float previous) {
        this.previous = 1 - offset - convertToRatio(previous);
        rawPre = previous;
        postInvalidate();
}

drawingLoi draw;
Paint back, end, black, drawBorder,blackBig;
Paint backInst, backMax, backPre;
float previous;

boolean flipped = false;
float threshole;
public void updateThresholdAndFlip(float threshole, boolean flipped) {
        this.threshole=threshole;
        this.flipped = flipped;
        if (flipped == false)
                realFlipThres = convertToRatio(max - threshole);
        else
                realFlipThres = convertToRatio(threshole);
}
int height;
public heightBar(Context context, AttributeSet attrs) {
        super(context, attrs);
        back = new Paint();
        back.setStyle(Paint.Style.FILL);
        back.setFlags(Paint.ANTI_ALIAS_FLAG);
        end = new Paint(back);
        backInst = new Paint(back);
        backMax = new Paint(back);
        backPre = new Paint(back);

        black = new Paint(back);
        black.setColor(Color.BLACK);
        black.setStyle(Paint.Style.STROKE);
        black.setTextSize(fontSize);
        blackBig=new Paint(black);
        blackBig.setTextSize(fontSize*1.5f);

        drawBorder = new Paint(black);
```

```
            drawBorder.setStrokeWidth(4);
            draw = new drawingLoi(end);
            Rect bounds = new Rect();
            String text = "Some random text";
            black.getTextBounds(text, 0, text.length(), bounds);
            height=bounds.height();
        }

        float percentOut = 0.2f;
        float wR = 0.8f;
        float wBar = 0.15f;
        int howDec = 3;
        float margin = 0.01f;
        final float heightIndicator = 0.2f;
        final float offset = heightIndicator / 2;
        float percentW = 0.5f;
        public float currentVal = 0f;
        public float currentMax = 0f;
        float max = 500f;
        float min = 0f;
        public int fontSize = 50;
        cPoint mot = new cPoint(0, 0);
        cPoint hai = new cPoint(1, 1);
        final float percentMaxW = 0.95f;
        final float percentCurW = 0.9f;
        final float percentPreW = 1f;

        public void setColor(int backColor, int endColor, int instColor, int preColor, int
maxColor) {

            end.setColor(endColor);
            back.setColor(backColor);
            backInst.setColor(instColor);
            backMax.setColor(maxColor);
            backPre.setColor(preColor);
        }

        float convertToRatio(float raw) {
            return (1 - 2*offset) * 1.0f * (raw - min) / (max - min);

        }

        void swap(float[] arr) {
            float tmp = arr[0];
            arr[0] = arr[2];
            arr[2] = tmp;
```

```
                tmp = arr[3];
                arr[3] = arr[4];
                arr[4] = tmp;

        }

        float realFlipThres;

        @Override
        protected void onDraw(Canvas canvas) {
                super.onDraw(canvas);

                draw.setAll(canvas, (float) getWidth(), (float) getHeight());
                // draw.drawRect(mot,hai, end);

                float cur = 1 - offset - convertToRatio(currentVal);
                float curMax = 1 - offset - convertToRatio(currentMax);

                int n = 5;
                float[] x = { margin +2*height/draw.w, wR - percentW * (wR - margin), wR -
margin,
                                wR - percentW * (wR - margin), margin +2*height/draw.w };

                float[] y = { cur - heightIndicator / 2, cur - heightIndicator / 2, cur, cur +
heightIndicator / 2,
                                cur + heightIndicator / 2 };
                float[] xp = new float[n];
                float[] yp = { previous - heightIndicator / 2, previous - heightIndicator / 2,
previous, previous + heightIndicator / 2,
                                previous + heightIndicator / 2 };

                for (int i = 0; i < n; i++) {
                        xp[i] = x[i];

                }
                xp[0] = x[0] - 2*height/draw.w;
                xp[4] = x[4] - 2*height/draw.w;

                float[] xm = new float[n];
                float[] ym = { curMax - heightIndicator / 2, curMax - heightIndicator / 2, curMax,
curMax + heightIndicator / 2,
                                curMax + heightIndicator / 2 };

                for (int i = 0; i < n; i++) {
                        xm[i] = x[i];
```

```
                }
                xm[0] = x[0]-height/draw.w;
                xm[4] = x[4]-height/draw.w;

                float[] xbox = { wR, wR + wBar, wR + wBar, wR };
                float[] ybox = { offset, offset, 1 - offset, 1 - offset };

                float[] xExtreme = { wR, wR + wBar, wR + wBar, wR };
                float leftWrite = x[2]-black.measureText("aa")/draw.w;
                int leftMost = 0;
                float[] yExtreme;
                if (flipped) {
                        for (int i = 0; i < xbox.length; i++) {
                                xbox[i] = (0.5f - xbox[i]) + 0.5f;
                                xExtreme[i] = (0.5f - xExtreme[i]) + 0.5f;

                        }
                        for (int i = 0; i < x.length; i++) {
                                x[i] = (0.5f - x[i]) + 0.5f;
                                xp[i] = (0.5f - xp[i]) + 0.5f;
                                xm[i] = (0.5f - xm[i]) + 0.5f;
                        }
                        swap(x);                swap(y);                swap(xm);
                swap(ym);               swap(xp);
                        swap(yp);

                        leftMost = 1;

                        leftWrite = x[1]-black.measureText("aa")/draw.w;
                        yExtreme=new float[] { 1 - offset - realFlipThres, 1 - offset -
realFlipThres, 1 - offset, 1 - offset };

                        draw.drawPolygon(back, xExtreme, yExtreme, 4);
                } else {
                        yExtreme =new float[] { offset, offset, offset + realFlipThres, offset +
realFlipThres };
                        draw.drawPolygon(back, xExtreme, yExtreme, 4);
                }
                draw.drawPolygon(drawBorder, xbox, ybox, 4);
                int[][] conner={{0,4},{2,1}};
                int choose=flipped?1:0;
                int tip=flipped?0:2;

                float corLab=wR+wBar;
                if (flipped) corLab = xbox[1];
```

```
                draw.drawText(label,corLab,
                                offset*0.6f, black,!flipped);
                draw.drawText(String.format("%.0f", max),flipped? xbox[1]+1.1f*wBar: wR-
0.1f*wBar,
                                ybox[1]+height/draw.w/2, black,!flipped);
//              flipped?"150":"120"
                draw.drawText(String.format("%.0f", threshole),flipped? xbox[1]+1.1f*wBar:
wR-0.1f*wBar,
                                yExtreme[flipped? 1:3]+height/draw.w/2, black,!flipped);
                draw.drawText("0",flipped? xbox[1]+1.1f*wBar: wR-0.1f*wBar,
                                ybox[3], black,!flipped);

                if (visiblePre) {
                        draw.drawPolygon(backPre, xp, yp, n);
                        draw.drawPolygon(black, xp, yp, n);
                        draw.drawTextLong("Prev\nSess", xp[leftMost]+(1-
choose)*height/draw.w
                                        +(-choose)*black.measureText("a")/draw.w, yp[tip] , black,
height, false);
                        draw.drawTextRotate("Prev", xp[conner[choose][0]],
yp[conner[choose][1]], black, height, flipped);

//                      draw.drawText("Sess", xp[leftMost]+(1-choose)*bounds.height()/draw.w,
yp[1] + (yp[3] - yp[1]) * 0.75f, black);
                        draw.drawTextLong(String.format("%." + howDec + "f", rawPre),
leftWrite, yp[tip], black, height, true);
                }
                draw.drawPolygon(backMax, xm, ym, n);
                draw.drawPolygon(black, xm, ym, n);
                draw.drawTextRotate("Max", xm[conner[choose][0]], ym[conner[choose][1]],
black, height, flipped);

                draw.drawTextLong("Sess\nMax", xm[leftMost]+(1-choose)*height/draw.w
                                +(-choose)*black.measureText("a")/draw.w, ym[tip], black, height,
false);
                draw.drawTextLong(String.format("%." + howDec + "f", currentMax ), leftWrite,
ym[tip], black,height,true);


                if (visibleInst) {
                        draw.drawPolygon(backInst, x, y, n);
                        draw.drawPolygon(black, x, y, n);
                        draw.drawTextRotate("Inst", x[conner[choose][0]], y[conner[choose][1]],
black, height, flipped);
                        draw.drawTextLong(String.format("%." + howDec + "f %s",
currentVal,extra_tmp), leftWrite, y[tip], black,height,true);
```

```
            }


        }

        String label = "";

        public void updateMinMaxPrecision(float min, float max, int decimal) {
            this.min = min;
            this.max = max;
            this.howDec = decimal;
        }

        public void setLabel(String label) {
            this.label = label;
        }

}
```

---

drawingLoi.java

```
package com.example.com.hdl.first_tutorial;

import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Paint.Align;
import android.graphics.Path;
import android.graphics.Rect;
import android.graphics.RectF;

public class drawingLoi {

        Canvas canvas;
        float w;
        float h;
        Paint defaultPaint;

        public drawingLoi(Paint defaultPaint) {

            this.defaultPaint = defaultPaint;
        }

        public void setAll(Canvas canvas, float w, float h) {
            this.w = w;
            this.h = h;
            this.canvas = canvas;
```

```
}

public void drawRect(cPoint p1, cPoint p2, Paint paint) {

	RectF a = new RectF(w * p1.x, h * p1.y, w * p2.x, h * p2.y);
	canvas.drawRect(a, paint);
}

public void drawCircle(cPoint point, float radius) {
	drawCircle(point, radius, defaultPaint);
}

public void drawPolygon(Paint stroke, float[] x, float[] y, int n) {
	int[] ax = new int[n];
	int[] ay = new int[n];
	Path p = new Path();

	for (int i = 0; i < n; i++) {
		if (x[i] > 1)
			x[i] = 1;
		if (x[i] < 0)
			x[i] = 0;
		if (y[i] > 1)
			y[i] = 1;
		if (y[i] < 0)
			y[i] = 0;
		ax[i] = (int) (x[i] * w);
		ay[i] = (int) (y[i] * h);
		if (i == 0)
			p.moveTo(ax[i], ay[i]);
		p.lineTo(ax[i], ay[i]);
	}
	p.lineTo(ax[0], ay[0]);
	// canvas.drawPath(p, defaultPaint);
	canvas.drawPath(p, stroke);
}

public void drawCircle(cPoint point, float radius, Paint paint) {
	canvas.drawCircle(point.x * w, point.y * h, radius, paint);
}

public void drawLine(cPoint mot, cPoint hai) {
	drawLine(mot, hai, defaultPaint);
}

public void drawLine(cPoint mot, cPoint hai, Paint paint) {
```

```
            canvas.drawLine(mot.x * w, mot.y * h, hai.x * w, hai.y * h, paint);
        }

        public void drawArc(cPoint p, float radius1, float start, float end, Paint paint) {
            float center_x, center_y;
            center_x = p.x * w;
            center_y = p.y * h;
            float radius = radius1 * w;
            final RectF oval = new RectF();
            oval.set(center_x - radius, center_y - radius, center_x + radius, center_y + radius);

            canvas.drawArc(oval, start, end, true, paint);
        }

        void drawTextLong(String text, float x, float y, Paint paint, int height, boolean right) {
            String[] sp = text.split("\n");
            if (sp.length == 2) {
                for (int i = 0; i < sp.length; i++)
                    canvas.drawText(sp[i], x * w, y * h + (i) * height, paint);
            } else {
                drawText(sp[0], x, y + height / 2 / h, paint, right);

            }

        }

        //
        void drawTextRotate(String text, float x, float y, Paint paint, int height, boolean flipped)
{
            canvas.save();
            canvas.rotate(flipped ? 90 : -90, x * w, y * h);
            // if (!flipped)
            canvas.drawText(text, x * w, y * h + height, paint);
            // else
            // canvas.drawText(text, x* w, y * h+bounds.height(), paint);
            // drawText(text, x+bounds.height()/h, y, paint,false);
            canvas.restore();
        }

        void drawText(String text, float x, float y, Paint paint) {
            drawText(text, x, y, paint, false);

        }

        void drawText(String text, float x, float y, Paint paint, boolean right) {// ,
            float off = right ? paint.measureText(text) : 0;
```

```
                canvas.drawText(text, x * w - off, y * h, paint);

        }

        public RectF convertRect(cPoint p1, cPoint p2) {
                return new RectF(w * p1.x, h * p1.y, w * p2.x, h * p2.y);
        }

        public Paint getPaintColor(int color) {
                Paint p = new Paint();
                p.setColor(color);
                ;
                return p;
        }
}
```

UserAdapter.java

```
package com.example.com.hdl.first_tutorial;

import java.util.ArrayList;

import com.example.com.hdl.first_tutorial.extraClass.Subject;
import com.example.com.hdl.first_tutorial.extraClass.visitInfo;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.TextView;

public class UsersAdapter extends ArrayAdapter<Subject> {
   public UsersAdapter(Context context, ArrayList<Subject> users) {
     super(context, 0, users);
   }

   @Override
   public View getView(int position, View convertView, ViewGroup parent) {
     // Get the data item for this position
     Subject user = getItem(position);
     // Check if an existing view is being reused, otherwise inflate the view
     if (convertView == null) {
       convertView = LayoutInflater.from(getContext()).inflate(R.layout.item_subject, parent,
false);
     }
```

```
        // Lookup view for data population
        TextView name = (TextView) convertView.findViewById(R.id.name);
        TextView age = (TextView) convertView.findViewById(R.id.age);
        // Populate the data into the template view using the data object
        name.setText("Subject: "+user.pseudo);
        age.setText("Age: "+user.age+"");
        // Return the completed view to render on screen
        return convertView;
    }
}
```

---

helper.java

```
package com.example.com.hdl.first_tutorial;

import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.Set;
import java.util.concurrent.Callable;

import com.example.com.hdl.first_tutorial.extraClass.someMethod;

import android.app.AlertDialog;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Context;
import android.content.DialogInterface;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Handler;
import android.os.ParcelUuid;
import android.util.Log;

public class helper {
        static String databaseName = "LoiStorage";

        public static void showMessage(Handler handler, final Context context, final String text,
final someMethod method) {
                handler.post(new Runnable() {

                        @Override
                        public void run() {
                                AlertDialog.Builder alertDialogBuilder = new
AlertDialog.Builder(context);
```

```
                alertDialogBuilder.setTitle("Info");


        alertDialogBuilder.setMessage(text).setCancelable(false).setPositiveButton("Yes",
                                new DialogInterface.OnClickListener() {
                                        public void onClick(DialogInterface dialog,
int id) {
                                                method.run();
                                        }
                                });
                        // }).setNegativeButton("No", new
                        // DialogInterface.OnClickListener() {
                        // public void onClick(DialogInterface dialog, int id) {
                        // // if this button is clicked, just close
                        // // the dialog box and do nothing
                        // dialog.cancel();
                        // }
                        // });

                        // create alert dialog

                        alertDialogBuilder.create().show();
                }

        });

    }
    public static void showYesNo(Handler handler, final Context context, final String text,
                final someMethod yesMethod, final someMethod noMethod) {
        handler.post(new Runnable() {

                @Override
                public void run() {
                        AlertDialog.Builder alertDialogBuilder = new
AlertDialog.Builder(context);
                        alertDialogBuilder.setTitle("Question");

                        alertDialogBuilder.setMessage(text).setPositiveButton("Yes", new
DialogInterface.OnClickListener() {
                                public void onClick(DialogInterface dialog, int id) {
                                        yesMethod.run();
                                }})
                        .setNegativeButton("No", new DialogInterface.OnClickListener() {
                                public void onClick(DialogInterface dialog, int id) {
                                        noMethod.run();
                                }}).show();
```

```
                }

            });

        }
        public static SQLiteDatabase openDatabase(Context v) {
            File test = v.getApplicationContext().getDatabasePath(databaseName);
            if (test.exists())
                return SQLiteDatabase.openDatabase(test.getPath(), null,
SQLiteDatabase.OPEN_READWRITE);
            else
                return null;
        }

        public static Object getMaxID(SQLiteDatabase lite, String sql, boolean isFloat) {
            Cursor resultSet = lite.rawQuery(sql, null);
            resultSet.moveToFirst();
            if (!resultSet.isAfterLast()) {
                if (isFloat)
                    return resultSet.getFloat(0);
                else
                    return resultSet.getInt(0);
            }
            return null;
        }

        private OutputStream outputStream;
        private InputStream inStream;

        private void init() throws IOException {
            BluetoothAdapter blueAdapter = BluetoothAdapter.getDefaultAdapter();
            if (blueAdapter != null) {
                if (blueAdapter.isEnabled()) {
                    Set<BluetoothDevice> bondedDevices =
blueAdapter.getBondedDevices();

                    if (bondedDevices.size() > 0) {
                        Object[] devices = (Object[]) bondedDevices.toArray();
                        BluetoothDevice device = (BluetoothDevice) devices[0];
                        ParcelUuid[] uuids = device.getUuids();
                        BluetoothSocket socket =
device.createRfcommSocketToServiceRecord(uuids[0].getUuid());
                        socket.connect();
                        outputStream = socket.getOutputStream();
                        inStream = socket.getInputStream();
```

```
                        }

                        Log.e("error", "No appropriate paired devices.");
                    } else {
                        Log.e("error", "Bluetooth is disabled.");
                    }
                }
            }

            public void write(String s) throws IOException {
                outputStream.write(s.getBytes());
            }

            public void run() {
                final int BUFFER_SIZE = 1024;
                byte[] buffer = new byte[BUFFER_SIZE];
                int bytes = 0;
                int b = BUFFER_SIZE;

                while (true) {
                    try {
                        bytes = inStream.read(buffer, bytes, BUFFER_SIZE - bytes);
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }
            }
        }
```

---

subjectSession.java

```
package com.example.com.hdl.first_tutorial;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.nio.channels.FileChannel;
import java.sql.Statement;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.LinkedList;
import java.util.Scanner;
import java.util.concurrent.Callable;

import com.example.com.hdl.first_tutorial.extraClass.Subject;
```

```
import com.example.com.hdl.first_tutorial.extraClass.someMethod;
import com.example.com.hdl.first_tutorial.extraClass.visitInfo;

import android.app.Dialog;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteException;
import android.os.Bundle;
import android.os.Environment;
import android.support.v7.app.ActionBar;
import android.support.v7.app.ActionBarActivity;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.BaseAdapter;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.NumberPicker;
import android.widget.TextView;
import android.widget.Toast;

@SuppressWarnings("deprecation")
public class subjectSession extends ActionBarActivity {
	ListView listView;
	ArrayList<visitInfo> listSess = new ArrayList<visitInfo>();
	MenuItem addSubjectMenu;

	SQLiteDatabase lite;

	@Override
	public boolean onCreateOptionsMenu(Menu menu) {
		MenuInflater inflater = getMenuInflater();
		inflater.inflate(R.menu.sessionmenu, menu);
		ActionBar actionBar = getSupportActionBar();
		actionBar.setDisplayOptions(0, ActionBar.DISPLAY_SHOW_TITLE);
		actionBar.setHomeButtonEnabled(true);
		actionBar.setDisplayHomeAsUpEnabled(true);
		return (super.onCreateOptionsMenu(menu));
```

```
	}

	public void notifyDataSet() {
		((BaseAdapter) listView.getAdapter()).notifyDataSetChanged();

	}

	void addSubject(int subjectID, String note, int age, char gender) {

		lite.execSQL(String.format("INSERT INTO subjectInfo
VALUES(%d,'%s','%s',%d,'%s');", subjectID,
				"subject" + subjectID, note, age, gender));

	}

	Subject sub = null;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.addsubject);
		Intent i = getIntent();
		if (i != null) {
			sub = (Subject) i.getSerializableExtra("passData");
			lite = helper.openDatabase(this);
		}

		listView = (ListView) findViewById(R.id.list);
		sessionAdapter adapter = new sessionAdapter(this, listSess);
		listView.setAdapter(adapter);

		// ListView Item Click Listener
		listView.setOnItemClickListener(new OnItemClickListener() {

			@Override
			public void onItemClick(AdapterView<?> parent, View view, int position,
long id) {
				Intent i = new Intent(getApplicationContext(), MainActivity.class);
				i.putExtra("visitData", listSess.get(position));
				i.putExtra("subData", sub);
				startActivity(i);

			}

		});
```

```
	}

	@Override
	public void onResume() {
		super.onResume(); // Always call the superclass method first

		showAll();
	}

	@Override
	public boolean onOptionsItemSelected(MenuItem item) {
		// Handle action bar item clicks here. The action bar will
		// automatically handle clicks on the Home/Up button, so long
		// as you specify a parent activity in AndroidManifest.xml.
		int id = item.getItemId();
		switch (id) {
		case (R.id.action_add_session):
			Intent i = new Intent(getApplicationContext(), MainActivity.class);
			i.putExtra("visitData", new visitInfo(-1));
			i.putExtra("subData", sub);
			startActivity(i);

			break;
		case android.R.id.home:
			this.finish();
			break;
		}
		return super.onOptionsItemSelected(item);
	}

	void showAll() {

		try {
			Cursor resultSet = lite.rawQuery("Select * from VisitInfo where
subjectID=" + sub.subjectID, null);

			listSess.clear();
			resultSet.moveToFirst();
			while (!resultSet.isAfterLast()) {
				Date date = null;
				try {
					date = new SimpleDateFormat("yyyyMMdd
hhmmss").parse(resultSet.getString(2));
				} catch (ParseException e) {
					// TODO Auto-generated catch block
					e.printStackTrace();
```

```
                        }

                        listSess.add(new visitInfo(resultSet.getInt(0), sub.subjectID, date,
resultSet.getFloat(3),
                                resultSet.getFloat(4))); // name
                        resultSet.moveToNext();
                    }
                    notifyDataSet();
                } catch (SQLiteException e) {
                    Toast.makeText(this, e.getMessage(), Toast.LENGTH_LONG).show();
                }
            }
}
```

SubjectManager.java

```
package com.example.com.hdl.first_tutorial;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.nio.channels.FileChannel;
import java.util.ArrayList;
import java.util.LinkedList;
import java.util.Scanner;
import java.util.concurrent.Callable;

import com.example.com.hdl.first_tutorial.extraClass.Subject;
import com.example.com.hdl.first_tutorial.extraClass.someMethod;

import android.app.Dialog;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteException;
import android.os.Bundle;
import android.os.Environment;
import android.support.v7.app.ActionBar;
import android.support.v7.app.ActionBarActivity;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.BaseAdapter;
```

```
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.NumberPicker;
import android.widget.Toast;

@SuppressWarnings("deprecation")
public class subjectManager extends ActionBarActivity {
	ListView listView;
	ArrayList<Subject> listSubject = new ArrayList<Subject>();
	MenuItem addSubjectMenu;

	SQLiteDatabase lite;

	@Override
	public boolean onCreateOptionsMenu(Menu menu) {
		MenuInflater inflater = getMenuInflater();
		inflater.inflate(R.menu.subjectmenu, menu);
		getSupportActionBar().setDisplayOptions(0,
ActionBar.DISPLAY_SHOW_TITLE);

		return (super.onCreateOptionsMenu(menu));

	}

	public void notifyDataSet() {
		((BaseAdapter) listView.getAdapter()).notifyDataSetChanged();

	}

	void addSubject(int subjectID, String note, int age, char gender) {

		lite.execSQL(String.format("INSERT INTO subjectInfo
VALUES(%d,'%s','%s',%d,'%s');", subjectID, subjectID, note,
				age, gender));

	}

	void showDetail(int index) {

		final int selection = index;
		final Dialog tmpDialog = new Dialog(subjectManager.this);
		tmpDialog.setContentView(R.layout.subjectdetail);

		final EditText txtname = (EditText) tmpDialog.findViewById(R.id.txtname);
```

```
                final NumberPicker txtage = (NumberPicker)
tmpDialog.findViewById(R.id.txtage);
                final CheckBox chkGen = (CheckBox) tmpDialog.findViewById(R.id.chkGen);
                tmpDialog.show();
                Button update = (Button) tmpDialog.findViewById(R.id.btnUpdate);
                Button delete = ((Button) tmpDialog.findViewById(R.id.btnDelete));
                txtage.setMinValue(1);
                txtage.setMaxValue(100);
                txtage.setValue(18);

                if (!lite.isOpen())
                        lite = helper.openDatabase(this);
                if (index == -1) {
                        // add new
                        tmpDialog.setTitle("Adding new...");
                        tmpDialog.show();
                       final int subjectID =(Integer) helper.getMaxID(lite, "Select max(subjectid)
from SubjectInfo", false) + 1;
                        txtname.setText(subjectID+"");

                        update.setText("Add");
                        update.setOnClickListener(new OnClickListener() {
                                @Override
                                public void onClick(View v) {
                                        // add new patient
//                                      if (txtname.getText().Trim().equals("")){
//                                              Toast.makeText(this, "N, duration)
//                                      }else{
                                        addSubject(subjectID, txtname.getText().toString(),
txtage.getValue(),
                                                        chkGen.isChecked() ? 'M' : 'F');
                                        showAll();
                                        tmpDialog.dismiss();
                                }
                        });

                        delete.setText("Cancel");
                        delete.setOnClickListener(new OnClickListener() {
                                @Override
                                public void onClick(View v) {
                                        tmpDialog.dismiss();
                                }
                        });
                } else {
                        txtname.setText(listSubject.get(selection).note);
                        txtage.setValue(listSubject.get(selection).age);
```

```
                        chkGen.setChecked(listSubject.get(selection).gender == 'M');

                        tmpDialog.setTitle("Information");
                        update.setOnClickListener(new OnClickListener() {
                                @Override
                                public void onClick(View v) {
                                        // update
                                        lite.execSQL(
                                                        String.format("update subjectinfo set
note='%s', age=%d,gender='%s' where subjectid=%d",
                                                                        txtname.getText().toString(),
txtage.getValue(),
                                                                        chkGen.isChecked() ? 'M' :
'F', listSubject.get(selection).subjectID));
                                        showAll();
                                        tmpDialog.dismiss();
                                }
                        });
                        delete.setOnClickListener(new OnClickListener() {
                                @Override
                                public void onClick(View v) {
                                        lite.execSQL(String.format("delete from subjectinfo where
subjectid=%d",
                                                        listSubject.get(selection).subjectID));
                                        lite.execSQL(String.format("delete from visitinfo where
subjectid=%d",
                                                        listSubject.get(selection).subjectID));
                                        lite.execSQL(String.format("delete from experiment where
subjectid=%d",
                                                        listSubject.get(selection).subjectID));

                                        showAll();
                                        tmpDialog.dismiss();

                                }
                        });
                }

        }

        @Override
        protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.addsubject);
                listView = (ListView) findViewById(R.id.list);
```

```
		UsersAdapter adapter = new UsersAdapter(this, listSubject);

		// Assign adapter to ListView
		listView.setAdapter(adapter);

		listView.setOnItemLongClickListener(new
AdapterView.OnItemLongClickListener() {

			public boolean onItemLongClick(AdapterView<?> arg0, View v, int
index, long arg3) {
				showDetail(index);
				return true;
			}
		});
		// ListView Item Click Listener
		listView.setOnItemClickListener(new OnItemClickListener() {

			@Override
			public void onItemClick(AdapterView<?> parent, View view, int position,
long id) {

				Intent i = new Intent(getApplicationContext(),
subjectSession.class);
				i.putExtra("passData", listSubject.get(position));
				startActivity(i);
			}

		});

		lite = helper.openDatabase(this);
		if (lite == null)
			createDB();
		else
			showAll();


	}

	void createDB() {
		lite = openOrCreateDatabase(helper.databaseName, MODE_PRIVATE, null);
		try {
			Scanner r = new
Scanner(this.getResources().openRawResource(R.raw.initialsql));
			LinkedList<String> arr = new LinkedList<String>();
			String current = "";
			while (r.hasNext()) {
```

```
                        String x = r.nextLine();
                        current += x;
                        if (x.lastIndexOf(";") > 0) {
                                arr.add(current);
                                current = "";
                        }

                }
                Object[] it = arr.toArray();
                for (int i = 0; i < it.length; i++) {
                        String x = (String) it[i];
                        lite.execSQL(x);
                }

        } catch (SQLiteException e) {
        }
        showAll();
        Toast.makeText(getBaseContext(), "Done Creating DB!",
Toast.LENGTH_LONG).show();
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();
        switch (id) {
        case (R.id.action_create_db):

                createDB();
                break;
        case (R.id.action_export_db):
                try {
                        File sd = Environment.getExternalStorageDirectory();

                        if (sd.canWrite()) {
                                String backupDBPath = "/cong1.db";
                                File currentDB =
getApplicationContext().getDatabasePath(helper.databaseName);
                                File backupDB = new File(sd, backupDBPath);

                                if (currentDB.exists()) {
                                        FileChannel src = new
FileInputStream(currentDB).getChannel();
```

```
                                        FileChannel dst = new
FileOutputStream(backupDB).getChannel();
                                        dst.transferFrom(src, 0, src.size());
                                        src.close();
                                        dst.close();
                                    } // if exists
                                    Toast.makeText(getBaseContext(), "Write to: " +
backupDB.toString(), Toast.LENGTH_LONG).show();

                                }
                            } catch (Exception e) {
                                Toast.makeText(getBaseContext(), e.toString(),
Toast.LENGTH_LONG).show();

                            }
                            break;
                        case (R.id.action_add_subject):
                            showDetail(-1);
                            break;

                        }
                        return super.onOptionsItemSelected(item);
                }

                void showAll() {

                        try {
                            Cursor resultSet = lite.rawQuery("Select * from SubjectInfo", null);
                            listSubject.clear();
                            resultSet.moveToFirst();
                            while (!resultSet.isAfterLast()) {
                                listSubject.add(new Subject(resultSet.getInt(0),
resultSet.getString(1), resultSet.getString(2),
                                        resultSet.getInt(3),
resultSet.getString(4).charAt(0))); // name
                                resultSet.moveToNext();
                            }
                            notifyDataSet();
                        } catch (SQLiteException e) {
                            Toast.makeText(this, e.getMessage(), Toast.LENGTH_LONG).show();
                        }
                }
}
```

---

sessionAdapter.java

```
package com.example.com.hdl.first_tutorial;
```

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;

import com.example.com.hdl.first_tutorial.extraClass.Subject;
import com.example.com.hdl.first_tutorial.extraClass.visitInfo;
import android.annotation.SuppressLint;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.TextView;

@SuppressLint("SimpleDateFormat")
public class sessionAdapter extends ArrayAdapter<visitInfo> {
	public sessionAdapter(Context context, ArrayList<visitInfo> users) {
		super(context, 0, users);
	}

	@Override
	public View getView(int position, View convertView, ViewGroup parent) {
		// Get the data item for this position
		visitInfo sess = getItem(position);
		// Check if an existing view is being reused, otherwise inflate the view
		if (convertView == null) {
			convertView =
LayoutInflater.from(getContext()).inflate(R.layout.item_subject, parent, false);
		}
		// Lookup view for data population
		TextView name = (TextView) convertView.findViewById(R.id.name);
		TextView age = (TextView) convertView.findViewById(R.id.age);
		// Populate the data into the template view using the data object
		name.setText("Session " + sess.visitId);
		age.setText(new SimpleDateFormat("MMM dd").format(sess.visitTime));

		// Return the completed view to render on screen
		return convertView;
	}
}
```

---

customCanvas.java

```
package com.example.com.hdl.first_tutorial;
```

```
import android.annotation.SuppressLint;


@SuppressLint("DrawAllocation")
public class customCanvas {

//
//	class cArc{
//		public float angle, value;
//		public cArc(float angle, float value){
//			this.angle=angle;
//			this.value=value;
//		}
//	}
//	Paint paint;
//	float w,h;
//	ArrayList<cArc> values=new ArrayList<cArc>();
//	Bitmap background=null;
//	int good= Color.rgb(51,148,46);
//	int medium=Color.rgb(144,148,46);
//	int bad=Color.rgb(148, 51,46);
//	float thres_GOOD=3f;
//	float thres_MEDIUM= 1f;
//	drawingLoi draw;
//	public customCanvas(Context context, AttributeSet attrs) {
//		super(context, attrs);
//		paint = new Paint();
//		paint.setStyle(Paint.Style.FILL);
//		paint.setColor(Color.MAGENTA);
//		paint.setStrokeWidth(8);
//		paint.setFlags(Paint.ANTI_ALIAS_FLAG);
//		draw=new drawingLoi(paint);
//
//
//	}
//	public void refreshBackground(){
//		background=openPicture();
//		this.postInvalidate();
//	}
//	Bitmap openPicture(){
//		Bitmap toDisk = null;
//		File file = new File(Environment.getExternalStorageDirectory().toString(),
//				"arun1.txt");
//		float[] metricArray=new float[MainActivity.MAX_ANGLE];
//		MainActivity.resetMetric(metricArray);
```

```
//                  if(file.exists())
//                  {
//                          BufferedReader br;
//
//                          try {
//                                  br = new BufferedReader(new FileReader(file));
//                                  String line;
//                                  while ((line = br.readLine()) != null) {
//                                          String[] parts = line.split(" ");
//
        metricArray[Integer.parseInt(parts[0])]=Float.parseFloat(parts[1]);
//
//
//                                  }
//                                  br.close() ;
//                          } catch (FileNotFoundException e) {
//                                  e.printStackTrace();
//                          } catch (NumberFormatException e) {
//                                  e.printStackTrace();
//                          } catch (IOException e) {
//                                  e.printStackTrace();
//                          }
//
//                          if (1==1)
//                                  try {
//
//                                          for (int i=0; i<MainActivity.MAX_ANGLE-1;i++)
//                                                  if (metricArray[i]<MainActivity.NON){
//                                                          int j;
//                                                          for (j=i+1;
j<MainActivity.MAX_ANGLE;j++)
//                                                                  if
(metricArray[j]<MainActivity.NON)
//                                                                          break;
//                                                          if (j!=MainActivity.MAX_ANGLE)
//                                                                  for (int k=i+1; k<j;k++)
//
        metricArray[k]=metricArray[i];
//
//                                                          i=j-1;
//
//
//                                                  }
//                                          values.clear();
//                                          for (int i=0; i<MainActivity.MAX_ANGLE;i++)
//                                                  if (metricArray[i]<MainActivity.NON)
```

```
//                                                  values.add(new cArc(
//                                                                  180-
getAngleFromRaw(i/100.0f),
//                                                                  metricArray[i]));
//
//                                      int w=400;
//                                      int h=w/2;
//                                      toDisk =
Bitmap.createBitmap(w,h,Bitmap.Config.ARGB_8888);
//                                      Canvas a=new Canvas(toDisk);
//                                      drawArcGrad(a, new cPoint(w,h), new cPoint(0.5f,0f), h,
values);
//                                      //                      drawArcGradMean(a, new
cPoint(w,h), new cPoint(0.5f,0f), h, values);
//                                      //                      String path =
Environment.getExternalStorageDirectory().toString();
//                                      //                      File file = new File(path,
"arun1.jpg"); // the File to save to
//                                      //
        toDisk.compress(Bitmap.CompressFormat.JPEG, 100, new FileOutputStream(file));
//
//                                  } catch (Exception ex) {
//                                      Log.d("Asd", "asd");
//
//                                  }
//              }
//
//              return toDisk;
//      }
//
//
//
//      void beep(){
//              ToneGenerator toneG = new
ToneGenerator(AudioManager.STREAM_NOTIFICATION,
//                                  (int)(ToneGenerator.MAX_VOLUME * 0.7));
//              toneG.startTone(ToneGenerator.TONE_PROP_BEEP, 200);
//      }
//      void drawArcGrad(Canvas canvas,cPoint dimen, cPoint center, float radius,
ArrayList<cArc> value ){
//              for (cArc point: value)
//              {
//                      cPoint bat=new cPoint(center.x*dimen.x, center.y*dimen.y);
//                      cPoint end=calculateEndPoint(bat, radius, point.angle);
//                      Paint paint=draw.getPaintColor( getColor(point.value));
//                      paint.setStrokeWidth(8);
```

```
//                      canvas.drawLine(bat.x,bat.y,end.x,end.y, paint);
//              }
//      }
//      float average(List<cArc> a){
//              float sum=0;
//              for (cArc b:a){
//                      sum+=b.value;
//              }
//              return sum/a.size();
//      }
//      int getColor(float value){
//              int color=bad;
//              if (value>thres_GOOD) color=good;
//              else if (value>thres_MEDIUM) color=medium;
//              return color;
//      }
//
//      //draw arc with clear separation
//      void drawArcGradMean(Canvas canvas,cPoint dimen, cPoint center,
//                      float radius, ArrayList<cArc> value ){
//              int length=value.size();
//              int start1= length/3;
//              int start2= length*2/3;
//
//              List<cArc> portion1= value.subList(0, start1);
//              List<cArc> portion2= value.subList(start1+1, start2);
//              List<cArc> portion3= value.subList(start2, length-1);
//              List<List<cArc>> mang=Arrays.asList(portion1, portion2, portion3);
//
//              int[] color={getColor(average(portion1)),
//                              getColor(average(portion2)),
//                              getColor(average(portion3))};
//              for (int i=0;i<3;i++){
//                      List<cArc> portion= mang.get(i);
//                      for (cArc point: portion){
//                              cPoint bat=new cPoint(center.x*dimen.x, center.y*dimen.y);
//                              cPoint end=calculateEndPoint(bat, radius, point.angle);
//                              Paint paint=new Paint();
//                              paint.setColor(color[i]);
//                              canvas.drawLine(bat.x,bat.y,end.x,end.y, paint);
//                      }
//              }
//
//      }
//
//      float angle=30f;
```

```
//	int frameRate=60;
//	float velocity=0.1f;
//	float raw=0f;
//	float force=0f;
//	float metric;

//	public float updateAngleAndForce(float angle1, float force){
//
//		this.raw=angle1;
//		angle=getAngleFromRaw(angle1);
//		this.force=force;
//		this.postInvalidate();
//
//		metric=(1-force)/MainActivity.clamp(raw, 0.1f, 1);
//		return metric;
//	}
//	cPoint calculateEndPoint(cPoint base, float length, float angle){
//		double x_, y_;
//		double ang_rad= angle*Math.PI/180;
//		x_=length*Math.cos(ang_rad);
//		y_=length*Math.sin(ang_rad);
//		return new cPoint((float) (base.x+x_),
//				(float) (base.y+y_));
//	}
//	@Override
//	protected void onDraw(Canvas canvas)
//	{
//		super.onDraw(canvas);
//
//		w = (float) getWidth();
//		h = (float) getHeight();
//
//		draw.setAll(canvas, w, h);
//		cPoint mot= new cPoint(0.05f,0.25f);
//		cPoint hai= new cPoint(0.2f,0.25f);
//
//		float width=0.55f;
//		cPoint origin= new cPoint(hai.x-width, hai.y);
//		cPoint endpoint=new cPoint(hai.x+width, hai.y+width);
//		if (background!=null)
//			canvas.drawBitmap(background,null,draw.convertRect(origin,
endpoint),null);
//
//		Paint lightP=new Paint();
//		lightP.setColor(Color.rgb(255,204,204));
//		Paint darkP=new Paint();
```

```
//                darkP.setColor(Color.BLACK);
//                darkP.setTextSize(30);
//
//                draw.drawCircle(mot, 10f);
//                draw.drawCircle( hai, 10f);
//                draw.drawLine(mot,hai);
//                cPoint ba=  calculateEndPoint(hai, 0.65f, 180-angle);
//                cPoint bon= new cPoint(ba.x+0.3f,ba.y+0.05f);
//                //                angle+=velocity;
//                //                if (angle>90 || angle<0) velocity=-velocity;
//                //                if (Math.abs(angle-70)<=10)
//                //                        beep();
//
//                //                int color=bad;
//                //                if (angle>thres_GOOD) color=good;
//                //                else if (angle>thres_MEDIUM) color=medium;
//
//
//                draw.drawLine(hai,ba);
//                draw.drawRect(ba,bon, draw.getPaintColor(getColor(raw)));
//                draw.drawText(String.format("   %2$.3f",
//                                force,raw) , ba.x, bon.y, darkP);
//                //small force, but can move big angle.
//
//                //                this.postInvalidateDelayed(1000/frameRate);
//        }
}
```

SubjectDetail.java

```
package com.example.com.hdl.first_tutorial;

import android.os.Bundle;
import android.support.v7.app.ActionBarActivity;

public class subjectDetail extends ActionBarActivity {
        @Override
        protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.subjectdetail);
        }
}
```